DECEMBER 1968

SYSTEM AND SOFTWARE SIMULATOR

VOLUME III

UNITED STATES ARMY
COMPUTER SYSTEMS SUPPORT
AND EVALUATION COMMAND
WASHINGTON, D.C. 20310

3200.8 (Att 1 to Encl 1)
Mar 7, 66

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Leo J. Cohen Associates
334 West State Street
Trenton, New Jersey

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
SYSTEM AND SOFTWARE SIMULATOR VOLUME III

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Working Paper

5. AUTHOR(S) *(First name, middle initial, last name)*
Leo J. Cohen

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1968	242	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAB09-68-C-0118	
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
This documentation has been assembled and released by the sponsoring activity	USA Computer Systems Support and Evaluation Command Washington, D. C.

13. ABSTRACT

The System and Software Simulator (S3) is a digital event simulator written in FORTRAN IV and designed to perform simulations of computer systems hardware and software and of the workload being applied to the system.

This and the other three volumes constitute the complete documentation available on S3. Volume I describes the inputs, outputs, methods and capabilities of S3. Volume II contains the flowcharts, narrative description of the flowcharts, layouts and descriptions of the tables utilized by S3. Volume III contains descriptions of the assembly language used for preparation of input to S3, of the macro capability of the assembler, and of the modifications made to S3 to provide additional output data. Volume IV is the program documentation on the internal workings of the assembler. It consists of table descriptions, flow charts and narratives, and file descriptions.

These volumes are a collection of documentation delivered under two separate contracts. They have not been edited and as such are considered working papers.

DD FORM 1 NOV 68 1473

Security Classification

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Simulation						
Computer Systems						
Operating Systems						
Evaluation						
Timing						
Dynamic Event Simulation						

Security Classification

WORKING PAPER

The documentation on the System and Software Simulator (S3) contained in this and the other three volumes is considered a working paper and no claims are made as to its accuracy. There has been no attempt to edit the information. Discrepancies and inconsistencies are known to exist.

This information is being released as a service to interested parties and to satisfy the numerous requests for information on S3.

The documentation of S3 is contained in four volumes. Volumes I and II are contract end items delivered under contract number DA-49-083 OSA-3306 and contain the technical descriptions of S3. Volume I describes the inputs, outputs, methods and capabilities of S3. Volume II contains the flowcharts, narrative description of the flowcharts, layouts and descriptions of the tables utilized by S3.

Volumes III and IV contain the documentation delivered as contract end items under contract number DAAB09-68-C-0118. Volume III contains descriptions of the assembly language used for preparation of input to S3, of the macro capability of the assembler, and of the modifications made to S3 to provide additional output data. Volume IV is the program documentation on the internal workings of the assembler. It consists of table descriptions, flow charts and narratives, and file descriptions.

TABLE OF CONTENTS

WORKING PAPER

SECTION I

USACSSEC S-3

EXTERNAL ASSEMBLER SPECIFICATIONS

Presented by
Leo J. Cohen Associates, Inc.

October 1, 1968

Revised October 24, 1968

334 West State Street
Trenton, New Jersey
08618
(609) 695-1488

110 N. Royal Street
Suite 305
Alexandria, Virginia
22314
(703) 548-0128

WORKING PAPER

Table of Contents

ALPHABETIC INDEX OF S-3 STATEMENTS

Format Rules for S-3 Input Statements

A. GENERAL

1. Labels, op-codes, and operands must not contain imbedded spaces.
2. Labels, op-codes, and operands must be followed by at least one space.
3. Comments may appear on any line of coding after the last operand and continue up to column 80.
4. Sequence numbers, if desired, are considered to be part of the comment.

B. LABELS

1. All labels must start in column 1.
2. Labels must not exceed 12 characters in length.
3. The first character of each label must be alphabetic.
4. Labels are not permitted on the JOB, OF, MEM-1, or MEM-2 statements.

C. OP-CODES

1. Op-codes may start anywhere except in column 1.
2. Op-codes must be spelled exactly as specified in this manual.

D. OPERANDS

1. The first operand must be preceeded by an op-code and at least one space but no more than 10 spaces.
2. Operands must not exceed 12 characters in length.
3. Operands must be separated by a comma.
4. No spaces may appear between operands but a space must appear after the last operand.
5. Operands may be continued onto the next line by following the last operand of the first line with a comma and a space and starting the first operand of the following lines in other than column 1.

Rules for Organization of Worker Routines

1. The first statement of every W/R must be the JOB statement.
2. The OF statements must immediately follow the JOB statement. There must be one OF statement for each file referenced within the W/R. If no files are referenced, then no OF statements should appear.
3. There must be a MEM-1 statement for each W/R. The MEM-1 statement must appear after the OF statements, if there are any, or after the JOB statement if there are no OF statements.
4. The last statement of every W/R must be a TERM statement.

CPU Definition

The CPU definition statement is used to supply the simulator with a description of the central processor unit or units which are to be used for this run.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	CPU-DEF	cpu-id (,LIB,CAT,NCAT)
(cpu hardware definition cards type 01 and 02, see appendix A)		
	CPU-END	

cpu-id

The "cpu-id" field must contain the name (limited to 6 characters) by which this CPU definition will be referenced by the configuration data. The use of this field is governed by the option chosen for the following field.

LIB

The LIB option specifies that the desired CPU definition may be found in the system library and that the CPU definition cards, types 01 and 02 are not included.

The CPU definition in the library must have the same name as the field "cpu-id". If the LIB option is selected, the CPU-END statement must be omitted.

CAT

The CAT option specifies that the desired CPU definition will be found on the next two cards, types 01 and 02 (see appendix A). In addition, this option will cause the CPU definitions to be entered into the library under the name supplied in the "cpu-id" field. If the CAT option is selected, the CPU-END statement may not be omitted.

NCAT

The NCAT option specifies that the desired CPU definition will be found on the next two cards, types 01 and 02 (see appendix A). However, the CPU definition will be used for the current simulation only, and will not be entered into the library. If the NCAT option is selected, the CPU-END statement may not be omitted.

Memory Definitions

The memory definition statement is used to supply the simulator with a description of the memory module or modules which are to be used for this run.

STATEMENT FORMAT		
NOT USED	OP-CD	OPERANDS
	MEM-DEF	mem-id (,LIB,CAT,NCAT)
(memory hardware definition card type 03, see appendix A)		
	MEM-END	

mem-id

The "mem-id" field must contain the name (limited to 6 characters) by which this memory definition will be referenced by the configuration data. The use of this field is governed by the option chosen for the following field.

LIB

The LIB option specifies that the desired memory definition may be found in the system library

and that the memory definition card, type 03 is not included in the input stream. The Memory Definition in the library must have the same name as the "mem-id" field. If the LIB option is selected the MEM-END statement must be omitted.

CAT

The CAT option specifies that the desired Memory definition will be found on the next card, type 03 (see appendix A). In addition, this option will cause the Memory Definition to be entered into the library under the name supplied in the "mem-id" field. If the CAT option is selected, the MEM-END statement may not be omitted.

NCAT

The NCAT option specifies that the desired Memory definition will be found on the next card, type 03 (see appendix A). However, the Memory definition will be used for the current simulation only and will not be entered into the library. If the NCAT option is selected, the MEM-END statement may not be omitted.

Channel Definitions

The channel definition statement is used to supply the simulator with a description of the channel or channels which are to be used for this run.

STATEMENT FORMAT		
NOT USED	OP-CD	OPERANDS
	CHAN-DEF	chan-id (,LIB,CAT,NCAT)
(channel hardware definition card type 04, see appendix A)		
	CHAN-END	

CHAN-ID

The CHAN-ID field must contain the name (limited to 6 characters) by which this channel definition will be referenced by the configuration data. The use of this field is governed by the option for the following field.

LIB

The LIB option specifies that the desired channel definition may be found in the system library and that the channel definition card, type 04 is not

included. The channel definition in the library must have the same name as the field "chan-id". If the LIB option is selected, the CHAN-END statement must be omitted.

CAT

The CAT option specifies that the desired channel definition will be found on the next card, type 04 (see appendix A). In addition, this option will cause the channel definition to be entered into the library under the name supplied in the "chan-id" field. If the CAT option is selected, the CHAN-END statement may not be omitted.

NCAT

The NCAT option specifies that the desired channel definition will be found on the next card, type 04 (see appendix A). However, the channel definition will be used for the current simulation only, and will not be entered into the library. If the NCAT option is selected, the CHAN-END statement may not be omitted.

Device Definitions

The device definition statement is used to supply the simulator with a description of the device or devices which are to be used for this run.

STATEMENT FORMAT		
NOT USED	OP-CD	OPERANDS
	DEV-DEF	dev-name (,LIB,CAT,NCAT)
(device hardware definition card type 05, appendix A)		
	DEV-END	

dev-id

The "dev-id" field must contain the name (limited to 6 characters) by which this device definition will be referenced by the configuration data. The use of this field is governed by the option chosen for the following fields.

LIB

The LIB option specifies that the desired device definition may be found in the system library and that the device definition card, type 05 is not

included. The device definition in the library must have the same name as the field "dev-id". If the LIB option is selected, the DEV-END statement must be omitted.

CAT

The CAT option specifies that the desired device definition will be found on the next card, type 05 (see appendix A). In addition, this option will cause the device definitions to be entered into the library under the name supplied in the "dev-id" field. If the CAT option is selected, the DEV-END statement may not be omitted.

NCAT

The NCAT option specifies that the desired device definition will be found on the next card, type 05 (see appendix A). However, the device definition will be used for the current simulation only, and will not be entered into the library. If the NCAT option is selected, the DEV-END statement may not be omitted.

Configuration Data

The Configuration Data supplied to the simulator specifies the organization of the hardware data which has already been defined. For example, the CPU statement shown below is used to specify the number of CPUs to be used in the current simulation. If there are to be two CPUs in the system then there must be two CPU statements. On the other hand, if both CPUs are identical, only one CPU-DEF statement was needed.

CONFIGURATION DATA

STATEMENT FORMAT		
NOT USED	OP-CD	OPERANDS
	CONFIG	(config-name) (,LIB,CAT)
	CPU	cpu-name, cpu-id
	.	
	.	
	MEM	mem-name, mem-id, cpu-name . . .
	.	
	.	
	CHANNEL	chan-name, chan-id, cpu-name . . .
	.	
	.	
	CONTROL	(IN,OUT,I/O), ctl-name, chan-name . . .
	.	
	.	
	DEVICE	(SEIZE,NOSEIZE), dev-name, dev-id,
		ctl-name . . .
	.	
	CONFIG-END	

CONFIG

The CONFIG statement indicates the beginning of the Configuration Data.

config-name

The "config-name" field is required, only if The Configuration Data is in the library, or is to be placed in the library. The usage of this field is defined by the option selected for the next field.

If the configuration data is neither in the library, nor to be placed in the library (NCAT option) no operands should be specified.

LIB

The LIB option specifies that the desired Configuration Data may be found in the system library and that the configuration data will not be found in the input stream. If the LIB option is specified, the CONFIG-END statement may not appear.

CAT

The CAT option specifies that the desired Configuration Data is to be placed into the system library. The Configuration Data must appear in the input stream and the CONFIG-END statement may not be omitted.

CPU

The CPU statement is used to specify the number and names of the CPUs required for the current simulation. There must be one CPU statement for each CPU in the system.

cpu-name

The "cpu-name" field must contain the name by which this CPU will be referenced during this simulation. CPUs are numbered in the order of their CPU statements.

cpu-id

The "cpu-id" field must contain the name (limited to 6 characters) of the CPU Definition which provides the operating characteristics of this CPU.

MEM

The MEM statement is used to specify the number and names of the memories required for the current simulation. There must be one MEM statement for each memory in the system.

mem-name

The "mem-name" field must contain the name by which this memory will be referenced during this simulation. Memories are numbered in the order of their MEM statements.

mem-id

The "mem-id" field must contain the name (limited to 6 characters) of the Memory Definition which provides the operating characteristics of this memory module.

cpu-name

The "cpu-name" fields, up to five, must contain the name of each CPU, as provided in the CPU statement, to which this memory is connected.

CHANNEL

The CHANNEL statement is used to specify the number and names of the channels required for the current simulation. There must be one CHANNEL statement for each channel in the system.

chan-name

The "chan-name" field must contain the name by which this channel will be referenced during this simulation. Channels are numbered in the order of their CHANNEL statement.

chan-id

The "chan-id" field must contain the name (limited to 6 characters) of the Channel Definition which provides the operating characteristics of this channel.

cpu-name

The "cpu-name" fields, up to five, must contain the name of each CPU, as provided in the CPU statement, to which this channel is connected.

CONTROL

The CONTROL statement is used to specify the number and names of the control units required for the current simulation. There must be one CONTROL statement for each control unit in the system.

IN

The IN operand specifies that this control unit is an input unit only.

OUT

The OUT operand specifies that this control unit is an output unit only.

I/O

The I/O operand specifies that this control unit is capable of performing both input and output operations.

The "ctl-name" field must contain the name by which this control unit will be referenced during this simulation. Control units are numbered in the order of their CONTROL statements.

chan-name

The "chan-name" field must contain the name of each channel, as provided in the CHANNEL statement, to which this control unit is connected.

DEVICE

The DEVICE statement is used to specify the number and names of the devices required for the current simulation. There must be one DEVICE statement for each device in the system.

SEIZE

The SEIZE operand specifies that this device may be seized by individual worker routines during the course of the simulation.

NOSEIZE

The NOSEIZE operand specifies that this device may not be seized by worker routines during this simulation.

dev-name

The "dev-name" field must contain the name by which this device will be referenced during this simulation.

dev-id

The "dev-id" field must contain the name (limited to 6 characters) of the Device Definition which provides the operating characteristics of the device.

ctl-name

The "ctl-name" field must contain the name of the control unit, as provided in the CONTROL, statement to which this control unit is connected.

TO-FROM Table Definition

The TO-FROM table is used to specify the amount of time required to move a random access device arm from one file to another. When files are assigned to random access units, they have relative locations on the device from 1 to the total number of files on the device. The user may then define the amount of time required to move from one file to another, or the average time required to move from one record in a file to another record in the same file.

STATEMENT FORMAT		
NOT USED	OP-CD	OPERANDS
	TF-DEF	dev-name, #-files
	TABLE	N11, N12, N13
	TABLE	N21, N22, N23
	TABLE	N31, N32, N33
	TF-END	

dev-name

The "dev-name" field must contain the name of the device to which the current TO-FROM definition applies. This name must be the same as the name supplied for the device in the configuration data.

#-files

The "#-files" field must contain the number of files which are assigned to the device.

TABLE

Each row in the TO-FROM table must begin with the operation code "TABLE".

N 11

The "N 11" operand must contain the amount of time in milliseconds required to get from file 1 to file 1 on the current device. Each operand must have a maximum of 5 decimal digits.

N 12

The "N 12" operand must contain the amount of time in milliseconds required to get from file 1 to file 2 on the current device.

The sample TO-FROM table illustrated above provides the required information for a device with 3 files. A TO-FROM table may be constructed for a device with up to 13 files.

QUEUE Definition

The QUEUE definition statement is used to supply the simulator with descriptions of the queues which will be used during the current simulation.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	Q-DEF	
	QUEUE	q-name, entries (,AT,IOT)(,FIFO,LIFO,PRI)
	.	
	.	
	Q-END	

q-name

The "q-name" field must contain the name by which this queue will be referenced during the current simulation.

entries

The "entries" field must contain a 1 to 4 decimal digit number specifying the maximum number of entries which this queue will be permitted to contain.

AT

If the AT operand is specified, the queue being defined may not contain I/O transaction items.

IOT

If the IOT operand is specified, the queue being defined may not contain available transactions. AT and IOT may not be specified for the same queue.

FIFO

If the FIFO operand is specified, the queue being defined will be organized as a first in, first out queue.

LIFO

If the LIFO operand is specified, the queue being defined will be organized as a last in, first out queue.

PRI

If the PRI operand is specified, the queue being defined will be organized in priority sequence.

REAL FILE Descriptions

The REAL FILE statement describes the real files which will be used during this simulation.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	FILE-DEF	
	RF	file-name,dev-name,rel-loc,buff-len,rec/buff,buff/file
	.	
	RFC	old-file-name,new-file-name(,dev-name)(,rel-loc)
	.	
	FILE-END	

file-name

The "file-name" field must contain the name by which this real file will be referenced by ORDINAL FILE statements.

dev-name

The "dev-name" field must contain the name of the device on which this file resides.

rel-loc

The "rel-loc" field must contain the relative location of this file on the device specified if there is more than one file on that device.

buff-len

The "buff-len" field must contain the length of the buffer, in characters, required to hold a physical record from this file.

rec/buff

The "rec/buff" field must contain the number of logical records contained in one physical record from this file.

buff/file

The "buff/file" field must contain the number of physical records contained in this file, i.e., file size.

REAL FILE COPY

In order to assist the user in describing a large number of files whose physical characteristics are identical, provision is made for copying the physical characteristics of a file already described, and giving the file a new name, device assignment and relative location. To perform this operation, the RFC statement is supplied.

old-file-name

The "old-file-name" field must contain the name of a file previously described by means of a RF statement. All physical information is copied from that file's description.

new-file-name

The "new-file-name" field must contain the name by which this copied file description is to be known.

dev-name

If the "new-file" is to reside on a different device from the "old-file", the "dev-name" field must contain the name of this device.

rel-loc

If the "new-file" is to have a different relative location from the "old-file", the "rel-loc" field must contain new relative locations.

LOAD CLASS

The LOAD CLASS statement allows the user to specify combinations of CPUs which are able to load programs for the system being simulated. For example, if program A can be loaded by CPUs 1 and 2, but program B may only be loaded by CPU2, then a load class of CPUs 1 and 2 and another load class consisting of CPU2 only must be defined. Program A would then be defined as being in load class 1 while program B is in load class 2.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	LC-DEF	
	LOAD	class-#, cpu-name (,cpu-name)
	LC-END	

class-#

The "class-#" must be a unique number from 1 to 15.

cpu-name

For each load class entry each of the CPUs in that entry must be named.

NOTE: If the current simulation uses only one CPU then the LOAD CLASS entry may be omitted.

RUN CLASS

The RUN CLASS statement allows the user to specify which CPU or CPUs may execute a program based upon the CPU which performed the load function. For example, in a system where there are two CPUs and a single memory shared by both of them, then any program loaded by either CPU may be executed by either CPU. However, if the two CPUs each had their own memory, although a program could be loaded by either CPU, it could only be executed by the CPU which loaded it.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	RC-DEF	
	RUN	loading-cpu-name,cpu-name (,cpu-name)
	. . .	
	RC-END	

load-cpu

The "load-cpu" field must contain the name of the CPU for which a RUN CLASS is being described. If there is more than one CPU in the current simulation, there must be one RUN statement for each CPU.

cpu-name

The "cpu-name" field must contain the name of each CPU capable of executing a program loaded by the CPU specified in the "load-cpu" field.

NOTE: If the current simulation uses only one CPU then the RUN CLASS statement may be omitted.

TABLE PRINTOUT CONTROL

The TABLE PRINTOUT CONTROL statement is used to specify which tables are to be printed at the end of each statistical interval.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	TAB-CTL	(ON,OFF)(,T1)(,T2)(, . . .)

ON

If the ON operand is specified, the normal condition for each table switch will be ON. Those tables which are not to be printed may be specified in subsequent operands.

OFF

If the OFF operand is specified, the normal condition for each table switch will be OFF. Those tables which are to be printed may be specified in subsequent operands.

TABLE NUMBER	TABLE NAME
T1	FUTURE EVENTS CHAIN
T2	CPU ITEM
T3	TRANSACTION ITEM
T4	I/O TRANSACTION ITEM
T5	WORKER ROUTINE BASE
T6	FILE VECTOR
T7	MEMORY
T8	PAGE
T9	LOAD CLASS
T10	RUN CLASS
T12	QUEUE
T13	QUEUE ENTRY
T14	FILE
T15	DEVICE SET
T16	AVAILABILITY
T17	CHANNEL
T18	CONTROL
T19	DEVICE CLASS
T20	GENERAL SIMULATION
T21	FUNCTION
T22	STATEMENT
T23	SWITCH
T24	TO-FROM
T25	MARK TIME

STATISTICS PRINTOUT CONTROL

The STATISTICS PRINTOUT CONTROL statement is used to specify which sets of statistics are to be printed at the end of each statistical interval.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	STAT-CTL	(ON,OFF)(ST1)(,ST2)(, . . .)

ON

If the ON operand is specified, the normal condition will be for all statistics to be printed. Those statistics which are not to be printed may be specified in subsequent operands.

OFF

If the OFF operand is specified, the normal condition will be for no statistics to be printed. Those statistics which are to be printed may be specified in subsequent operands.

STATISTICS NUMBER	STATISTICS NAME
ST1	QUEUE
ST2	MEMORY
ST3	FILE
ST4	DEVICE
ST5	CONTROL UNIT
ST6	CHANNEL
ST7	CPU
ST8	QUEUE ANALYSIS
STAT	W/R

STATISTICS INTERVAL CONTROL

The STATISTICS INTERVAL CONTROL statement is used to control the interval between printing of statistics and the number of such intervals which are to be simulated.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	STATISTICS	minutes,count

minutes

The "minutes" field must contain the number of minutes in a single simulation interval.

count

The "count" field must contain the total number of intervals to be simulated.

GLOBAL EQUATE

The GLOBAL EQUATE statement assigns a value to a label which will be recognized in any routine assembled during the current simulation. All global equate statements must appear before the first job statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
symbol	GEQU	value

symbol

The "symbol" field must begin with an alphabetic character and must be limited to 12 or less characters. When this symbol appears in any other statement, it will be replaced with the value specified as the operand for this statement.

value

The "value" field must contain a numeric value of 4 or less decimal digits. This is the value which will be used to replace the symbol described above.

LOCAL EQUATE

The LOCAL EQUATE statement assigns a value to a label which will be recognized only in the routine currently being assembled. A local equate statement must appear after the JOB statement and before the END statement of the routine in which it is to be used.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
symbol	LEQU	value

symbol

The "symbol" field must begin with an alphabetic character and is limited to 12 or less characters. When this symbol appears in any other statement of the current routine, it will be replaced with the value specified in the operand of this statement.

value

The "value" field must contain a numeric value of 4 or less decimal digits.

JOB

The JOB statement defines a worker routine. In addition, it provides the simulator with certain information about the worker routine.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	JOB	job-name, (OS,PW,BUS,SCI), (REENT,NREENT)

job-name

The "job-name" field must contain the unique name of the current JOB.

OS

The OS operand informs the simulator that the current JOB is an operating system.

PW

The PW operand informs the simulator that the current JOB is a primary worker.

BUS

The BUS operand informs the simulator that the current JOB is a commercial worker routine.

SCI

The SCI operand informs the simulator that the current JOB is a scientific worker routine.

REENT

The REENT operand informs the simulator that the current JOB is reentrant.

NREENT

The NREENT operand informs the simulator that the current JOB is not reentrant.

ORDINAL FILE

The ORDINAL FILE statement describes the files to be used by the current JOB. In addition, it describes which files and devices must be seized before the current JOB can be executed.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	OF	of-name, rf-name, buff-no, (NOSEIZE, SEIZE-DEV, SEIZE-FILE)

of-name

The "of-name" field specifies the name by which the file is referenced within the current JOB.

rf-name

The "rf-name" field specifies the name of the real file which is to be used when the OF-NAME is referenced.

buff-no

The "buff-no" field must contain the number of buffers which are to be allocated for use by the file.

NOSEIZE

The NOSEIZE operand specifies that neither the file nor the device must be seized for I/O operation referencing this file.

SEIZE-DEV

The SEIZE-DEV operand specifies that the device on which the file resides must be seized before this JOB may begin.

SEIZE-FILE

The SEIZE-FILE operand specifies that this file must be seized before the JOB may being.

MEMORY-1

The MEMORY-1 statement provides the user with a simple method for describing the amount of memory which will be required to run the current JOB.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MEM-1	inst, char

inst

The "inst" field specifies the number of instructions required for the current JOB. This operand must be in the format DDDE.

char

The "char" field specifies the number of characters of storage required for the current JOB. This operand must be in the format DDDE.

Format DDDE is interpreted as a number in the range from 1 to 999 with an exponent in the range from 0 to 9.

MEMORY-2

The MEMORY-2 statement provides the user with a detailed mechanism for describing the amount of data storage required by the current JOB.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MEM-2	dec-field, dec-len, bin-fields, bin-len,
		float-fields, float-len

dec-fields

The "dec-fields" field must contain the number of decimal fields required by the current JOB.

dec-len

The "dec-len" field must contain the length, in decimal digits, of the dec-fields.

bin-fields

The "bin-fields" field must contain the number of binary fields required by the current JOB.

bin-len

The "bin-len" field must contain the length, in decimal digits of the "bin-fields" field.

float-fields

The "float-fields" field must contain the number of floating-point fields required by the current JOB.

float-len

The "float-len" field must contain the length, in decimal digits, of the "float-fields" field.

GENERATE

The GENERATE statement allows the user to specify the frequency and total number of transactions which are to be generated for the current JOB.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	GEN	interval, (FIX,POISS), limit, priority

interval

The "interval" field specifies the interval in seconds between the generation of transactions for the JOB. This operand must be in the formal DDDE.

Format DDDE is interpreted as a number in the range from 1 to 999 with an exponent in the range from 0 to 99.

FIX

The FIX operand specifies that the interval between the generation of transactions for this JOB is to be considered as a fixed value.

POISS

The POISS operand specifies that the interval between the generation of transactions for this JOB is to be modified by a poisson distribution. This enables the user to vary the interval of generation.

limit

The "limit" field must contain the total number of transactions which are to be generated for this JOB.

priority

The "priority" field must contain the priority which is to be associated with this JOB. The priority must be in the range of 1 to 99 with 1 as the highest priority and 99 the lowest.

TRANSFER

The TRANSFER statement allows the user to alter the normal sequence of statements to be executed by the simulator. Normally, statements are executed sequentially one right after another. Each time a statement is executed, the next sequential instruction counter (NSI) is incremented to the address of the next statement to be executed. When a TRANSFER statement is executed, the NSI is replaced with the address of the statement specified by LOC.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	TRA	loc

loc

The "loc" field must contain the label of the next statement to be executed.

TRANSFER ON PROBABILITY

The TRANSFER ON PROBABILITY statement is much like the TRANSFER statement. However, instead of automatically changing the next sequential instruction counter to the address of the statement specified by LOC, a draw is first made from a pseudo-random number generator. Since the random number generator provides numbers in the range from 0 to 99, if the random number provided is less than percentage probability of transfer a transfer will occur. If the random number generated is equal to or greater than the percentage probability of transfer no transfer will occur and the next sequential instruction will be executed. If a transfer does occur, the next statement to be executed will be found at the address specified by the field LOC.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	TRA-P	PERCENT,LOC

percent

The "percent" field must contain a number within the range 0 to 100. This field specifies the probability of making a transfer to the address specified by LOC. It should be realized that if the percent field is set equal to 0 the instruction effectively acts as a no-op. If the percentage probability for transfer is equal to 100 then the instruction effectively acts as an unconditional transfer.

loc

The "loc" field must contain the label of the statement to be executed if transfer does occur.

READ

The READ statement causes the simulator to generate an interrupt into the operating system. This means that the current operating transaction (COT) will become the available transaction (AT) and an operating system transaction will be generated as COT. The next sequential instruction counter for the COT will then be set to the interrupt address specified for the READ interrupt. It will then become the responsibility of the operating system, using the statements provided, to handle the input/output operation requested.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	READ	of-name

of-name

The of-name operand specifies the name of the ordinal file for which this READ operation is to occur.

It is important to note that there must be an ordinal file card for every ordinal file referenced in a worker routine.

When this statement is used within the operating system, the READ interrupt must be enabled.

WRITE

The WRITE statement causes the simulator to generate a WRITE interrupt into the operating system. This means that the current operating transaction (COT) will become the available transaction (AT) and an operating system transaction will be generated as COT. The next sequential instruction counter for the COT will then be set to the interrupt address for the WRITE interrupt. It will then become the responsibility of the operating system, using the statements provided, to handle the I/O operation requested.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	WRITE	of-name

of-name

The "of-name" operand specifies the name of the ordinal file for which this WRITE operation is to occur. It is important to note that there must

be an ordinal file card for every ordinal file referenced within a worker routine.

When this statement is used within an operating system the WRITE interrupt must be enabled.

FUNCTION

The FUNCTION statement causes the simulator to generate a FUNCTION interrupt into the operating system. This means that the current operating transaction (COT) will become the available transaction (AT) and an operating system transaction item will be generated as the new current operating transaction (COT). The next sequential instruction counter for the COT will then be set to the interrupt address for the FUNCTION interrupt.

However, unlike the READ or WRITE statements the FUNCTION statement will also cause the immediate generation of an I/O transaction item. The duration of the I/O operation which may be initiated by the operating system as a result of the FUNCTION interrupt is governed by the entries in the system function table. Each entry in the system function table specifies the length of time the channel, control unit, and device are to be busy with this I/O operation.

STATEMENT FORMAT		
LABEL	OP-CD	
	FUNCTN	fno

fno

The " fno" field specifies the number of the entry in the system function table which is to be referenced by the I/O advance statement in the operating system. Function numbers are not like ordinal files in that they must be assigned uniquely across all worker routines within a given simulation. For this reason, the FUNCTION statement is not considered a normal IO operation and should be used only for I/O operations which cannot be represented using READ or WRITE statements.

When this statement is used within an operating system the FUNCTION interrupt must be enabled.

END OF FILE

The END OF FILE statement allows the worker routine to check for an end of file on any one of the ordinal files being used. The simulator keeps a count of the total number of I/O operations performed on each ordinal file. In addition, the number of blocks in each file may be specified in the file description system parameter card. When this statement is executed, a comparison is made between the block count for the real file and the number of I/O operations performed on that file by the current transaction. If the number of IO operations performed is equal to the number of blocks in the file being referenced, a transfer will be made to the address in the "loc" field of this statement. Otherwise the next sequential instruction will be executed.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	EOF	of-name, loc

file

The "file" field specifies the name of the ordinal file for which the END OF FILE test is to be executed. It is important to note that there must be an ordinal file card for every ordinal file referenced in a worker routine.

loc

The "loc" field must contain the label of the statement to be executed if an end of file condition is found to exist.

The END OF FILE statement may not be used on a file which has been opened more than once by the same transaction item.

SUBROUTINE

The SUBROUTINE statement allows the user to execute any segment of code within a worker routine as a closed subroutine. When the SUBROUTINE statement is encountered the next sequential instruction counter (NSI) is stored into one of the three fields provided in the transaction item for this purpose. The NSI is then set equal to the address of the subroutine to be executed. The end of a subroutine is indicated by an EXIT statement which will return control to the statement after the SUBROUTINE statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	SUB	loc

loc

The "loc" field must contain the label of the first statement in the subroutine to be executed.

ERROR CONDITIONS

A worker routine is permitted a maximum of three nested subroutine calls at any given time. If a fourth nested subroutine is encountered during a simulation an appropriate diagnostic will be issued and simulation will terminate.

EXIT

The EXIT statement is used to indicate the end of a subroutine. When an EXIT statement is encountered the next sequential instruction counter for the current operating transaction will be set equal to the statement immediately following the last SUBROUTINE statement executed.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	EXIT	

The EXIT statement has no operands.

ERROR CONDITION

If an EXIT statement is encountered when there are no active subroutines for the current operating transaction an appropriate diagnostic will be issued and simulation will terminate.

LOOP

The LOOP statement allows the user to execute a segment of code a specified number of times. Once the specified segment of code has been executed the number of times specified, the LOOP statement will be ignored and the statement following the LOOP statement will be executed.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	LOOP	count, loc

count

The "count" field specifies the number of times a transfer to the location specified in the "loc" field should occur.

loc

The "loc" field must contain the label of the first statement in the segment of code to be executed repeatedly. Each time the LOOP statement is executed

a LOOP counter will be decremented and a transfer to the statement with label "loc" will occur. Once the LOOP counter has reached 0 the statement after the LOOP statement will be executed next. Noramlly the statement with label "loc" should precede the LOOP statement.

ERROR CONDITIONS

A worker routine is permitted a maximum of three nested loops at any given time. If a fourth LOOP statement is encountered while all three LOOP counters contain a value, an appropriate diagnostic will be issued and simulation will be terminated.

MOVE

The MOVE statement allows the user to represent the amount of time required to perform a MOVE operation. A simulated advance time will be calculated by dividing the number of characters to be moved by the number of characters in a logical data unit and multiplying the result by the amount of tim required to move a single logical data unit.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MOVE	char char

The "char" field specifies the number of characters to be moved as a memory to memory transfer of data.

MOVE AND EDIT

The MOVE AND EDIT statement allows the user to simulate the amount of time required to perform a MOVE AND EDIT operation. A simulated advance time is calculated by dividing the number of characters to be moved and edited by the amount of time to move and edit a single character.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MOVE-E	char

char

The "char" field specifies the number of characters to be moved and edited within a memory by this MOVE AND EDIT operation.

COMPUTE

The COMPUTE instruction allows the user to represent the execution of a specified number of instructions. The amount of time required to execute the number of instructions specified will be added to the total CPU time used by the current operating transaction.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	COMPUT	inst,(mark)

inst

The "inst" field specifies the number of instructions to be simulated. This field will be multiplied by the average instruction execution time to develop an advance time for this operation. The simulator will then draw a random entry from the Poisson Distribution table and multiply this entry times the advance time calculated.

mark

The "mark" field allows the user to total all advance times for selected compute statements. This field must contain the number of the mark time accumulator into which the advance time for this statement will be added.

MATH

The MATH statement provides the ability to represent precisely the time required to perform arithmetic operations. The user specifies the number of ADD, MULTIPLY, and DIVIDE operations to be represented by this MATH statement. The simulator then uses the add, multiply, and divide times specified in the CPU definition to calculate the advance time required. In the MATH statement, unlike the COMPUTE statement, the advance time computed is not weighted by a Poisson Distribution.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MATH	add, mult, div

add

The "add" field specifies the number of additions or subtractions to be represented by this statement.

mult

The "mult" field specifies the number of multiply operations to be represented by this statement.

div

The "div" field specifies the number of divide operations to be represented by this statement.

OPEN

The OPEN statement is used to initialize the buffer and record counts for an ordinal file before the ordinal file is used for other I/O operations. With the exception of the operating system every worker routine must OPEN an ordinal file before it may be referenced by a READ, WRITE, or PRINT statement. The OPEN statement generates an immediate OPEN interrupt and an I/O transaction item. If the OPEN operation specified is for an input file, the I/O transaction item will represent the time required to fill all available buffers for the file. If the OPEN operation specified is for an output file, the IO transaction item will require no time to execute and will merely initialize the buffer and record counts.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	OPEN	of-name, (INPUT,OUTPUT)

of-name

The "of-name" field specifies the ordinal file name of the file which is to be opened.

INPUT

The INPUT operand specifies the ordinal file to be opened is an input file.

OUTPUT

The OUTPUT operand specifies the ordinal file to be opened is an output file.

CLOSE

The CLOSE statement is used to release devices which have been seized by a PERIPHERAL statement. In addition, for tape files only, the CLOSE operation may be used to initiate a rewind operation which will keep the device busy for the amount of time required to complete a REWIND operation, as specified in the device definition.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	CLOSE	of-name, (REWIND,NOREWIND)

of-name

The "of-name" field specifies the ordinal file name which is to be closed by this statement.

REWIND

The REWIND operand specifies that if the ordinal file being closed resides on a tape unit, a rewind operation is to be performed.

NOREWIND

The NOREWIND operand specifies that even if the file being closed resides on a tape device no rewind operation is to be performed.

TERMINATE

The TERMINATE statement generates an immediate TERMINATE interrupt into the operating system. This interrupt is used to inform the operating system that a transaction item has completed all requirements. In addition the TERMINATE statement may create a new transaction for another worker routine. This facility is useful when worker routines must be executed in a specific sequence. In addition to being the last logical statement in a worker routine, the TERMINATE statement must also be the last physical statement.

If more than a single TERMINATE statement is required, they must be adjacent and must be the last statements in the worker routine.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	TERM	(job-name)

job-name

The "job-name" field, if used, specifies the name of the worker routine for which a new transaction will be generated immediately. This allows the termination of one worker routine transaction to trigger the generation of a transaction for another worker routine.

PRINT

The PRINT statement allows the user to represent the time required for a printer to space one or more lines after printing. In every other respect it is exactly the same as the WRITE statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	PRINT	of-name, spaces ..

of-name

The "of-name" field specifies the name of the ordinal file to which this PRINT operation is directed.

spaces

The "spaces" field specifies the number of space operations to be performed after the PRINT is complete. The number supplied for this operand is multiplied by the amount of time required to perform a single space and the total time calculated is added to the amount of time the device on which this ordinal file resides, will be busy.

CALL

The CALL statement provides the ability to inform the operating system that another worker routine's execution is required by the current operating transaction. This statement stores the called worker routine's name and generates an immediate interrupt into the operating system. This statement also allows the user to specify whether the current job is to be deferred until the newly called job is complete or whether the current job may continue processing with the newly created job.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	CALL	job-name, (DLY,NODLY)

job-name

The "job-name" field specifies the name of the worker routine to be called by this statement.

DLY

The DLY operand specifies the current operating transaction is to be deferred until the newly called worker routine has been completed.

NODLY

The NODLY operand specifies that the current operating transaction is not to be deferred until the newly created transaction is complete but rather that the current transaction may compete with the newly created transaction item for control of the CPU.

EXEC

The EXEC statement provides the ability to generate an interrupt into the operating system for the current CPU. Although the user may specify the generation of any one of the twenty interrupts available to its CPU, the generation of an open-close interrupt for example will not cause the creation of an I/O transaction item. Therefore, if the operating system requires an I/O transaction item for some interrupt, that interrupt should not be generated by an EXEC statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	EXEC	int-name

int-name

The "int-name" field specifies the name of the interrupt which is to be generated by this statement.

MEMORY

The MEMORY statement examines the memory map of the object system to determine whether there is sufficient memory to load a transaction item within the range of memories specified. The transaction item for which the memory test is made may be either the available transaction (AT) or it may be a transaction resident on a specified queue. The possible results from the memory test are as follows: 1. There is not enough memory in the object system. 2. A PACK operation would be required before the transaction item can be loaded. 3. There is sufficient memory in the object system to load the transaction item.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MEMORY	mem-name 1, mem-name 2 (,q-name)
	NSI	
	NSI+1	
	NSI+2	

mem-name 1

The "mem-name 1" field specifies the name of the first memory in the range to be tested by the memory statement.

mem-name 2

The "mem-name 2" field specifies the name of the last memory in the memory range to be tested by this statement.

q-name

If the "queue-name" field is omitted from the MEMORY statement, the transaction for which the MEMORY test will be made must be the available transaction (AT).

If the "queue-name" field is supplied for the MEMORY statement, the transaction item for which the MEMORY test will be made must be on the specified queue. The specific item on the queue for which the memory test will be made may be adjusted by use of the EXAMINE statements.

NSI

The NSI statement will be executed if there is not enough memory in the range specified to load the transaction item specified.

NSI+1

The NSI+1 statement will be executed if a PACK statement must be performed before there will be sufficient memory in the range specified to load the transaction item specified.

NSI+2

The NSI+2 statement will be executed if there is sufficient memory available to load the transaction item specified.

ALLOCATE

The ALLOCATE statement is used in conjunction with the MEMORY statement. Once the MEMORY statement has determined that there is sufficient memory available to load a transaction item the ALLOCATE statement may be used to assign the storage to the transaction item. It is important to note that attempting to perform an ALLOCATE statement when sufficient memory is not available will terminate the simulation. In addition the memory range specified for the ALLOCATE statement must be exactly the same as for the corresponding MEMORY statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	ALLOC	mem-name 1, mem-name 2

mem-name 1

The "mem-name 1" field specifies the name of the first memory in the range of memories to be used for this allocation.

mem-name 2

The "mem-name 2" field specifies the name of the last memory in the range of memories to be used for this allocation.

DEALLOCATE

The DEALLOCATE statement examines the object memory map and releases all memory assigned to the current available transaction (AT). However, if the worker routine being used by the current available transaction has been specified as being reentrant and there is another active transaction item associated with this worker routine only the data and I/O storage are released, instruction storage remains allocated. For a reentrant worker routine of this type, instruction storage is released when a DEALLOCATE statement is executed for the last active transaction item utilizing that worker routine.

The DEALLOCATE statement is normally followed by a DESTROY statement to eliminate the terminated transaction item from the system.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	DALLOC	

There are no operands associated with the DEALLOCATE statement.

ERROR CONDITIONS

Any attempt to execute a RETURN to an available transaction which has no memory allocated to it will generate a diagnostic and terminate the simulation.

PACK

The PACK statement provides the ability to reorganize the contents of a range of memories. The PACK statement moves all pages which are currently assigned to the low end of the memory range thereby leaving all free pages at the high end of the memory range. This allows an operating system to represent the dynamic reorganization of memory when a transaction item requires more contiguous storage than is currently available.

For an operating system which provides dynamic compacting of memory when necessary, the PACK statement is normally coded at the NSI+1 exit line from the MEMORY statement. This statement would then be followed by an ALLOCATE statement as the NSI+2 exit line. For an operating system which automatically reorganizes memory whenever a transaction item has its memory deallocated, the PACK statement is normally coded immediately following the DEALLOCATE statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	PACK	mem-name 1, mem-name 2

mem-name 1

The "mem-name 1" field specifies the first memory in the range of memories for which the PACK is to be performed.

mem-name 2

The "mem-name 2" field specifies the name of the last memory in the range for which the PACK is to be performed.

EXAMINE FIRST

The EXAMINE FIRST statement sets the queue pointer to the first entry in the specified queue. A test is then performed to determine if there is an item in the first entry. The EXAMINE FIRST statement does not alter the contents of the queue in any way.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	EXAM-F	q-name
	NSI	
	NSI+1	

q-name

The "q-name" field specifies the name of the queue for which the EXAMINE FIRST operation is to be performed.

NSI

The NSI statement will be executed if there is no item in the first entry of the specified queue.

NSI+1

The NSI+1 statement will be executed if there is an item in the first entry of the queue specified.

EXAMINE NEXT

The EXAMINE NEXT statement alters the queue pointer for the specified queue by one position. The EXAMINE NEXT statement must be preceded by an EXAMINE FIRST or an EXAMINE LAST statement for the same queue. If this statement was preceded by an EXAMINE FIRST statement, then the queue pointer is altered to look at the next item down the queue. If this statement was preceded by an EXAMINE LAST statement then the queue pointer is altered to look at the next item up the queue. Once the queue pointer has been altered a test is made to determine if the new entry for which the queue pointer is set contains an item or not. The EXAMINE NEXT statement does not alter the contents of the queue in any way.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	EXAM-N	q-name
	NSI	
	NSI+1	

q-name

The "q-name" field specifies the name of the queue for which the EXAMINE NEXT statement is to be executed.

NSI

The NSI statement will be executed if the current queue entry does not contain an item.

NSI+1

The NSI+1 statement will be executed if the current entry for the queue specified does contain an item.

EXAMINE LAST

The EXAMINE LAST statement sets the queue pointer to the last entry, if any, in the queue specified. A test is then performed to determine if there is an item in the queue. The EXAMINE LAST statement does not alter the contents of the queue in any way.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	EXAM-L	q-name
	NSI	
	NSI+1	

q-name

The "q-name" field specifies the name of the queue for which the EXAMINE LAST statement will be executed.

NSI

The NSI statement will be executed if there are no items on the queue for which the EXAMINE LAST statement was executed.

NSI+1

The NSI+1 statement will be executed if there is an item available in the queue for which the EXAMINE LAST statement was executed.

PLACE

The PLACE statement provides the ability to put the available transaction word or the I/O transaction word specified onto any selected queue. The manner in which the place is performed is determined by the queueing method selected when defining the queue. If queueing is by priority, the current item will be placed after any other items in the queue which have the same priority level. If the queueing method selected is first-in-first-out the current item will be placed ahead of any other item on the queue. Once the PLACE operation has been performed the queue pointer is set equal to the beginning of the queue, as if an EXAMINE FIRST had been executed.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	PLACE	(AT,IOT),q-name

AT

The AT operand specifies that the transaction word to be placed will be found in the AT word for the current operating CPU.

IOT

The IOT operand specifies that the transaction item to be placed will be found in the IOT word for the current operating CPU.

q-name

The "q-name" field specifies the name of the queue for which the place operation is to be performed.

ERROR CONDITIONS

Any attempt to place a zero transaction word on a queue or to place a transaction word on a queue which is full will cause the generation of an error message and the termination of the simulation.

SELECT

The SELECT statement provides the capability to remove a transaction word from a queue and place it into either the AT or the IOT word for the current operating CPU. The item to be selected from the specified queue is determined by the queue pointer for that queue. As has been previously described, the queue pointer may be manipulated by the EXAMINE statements. When the SELECT statement is executed, the AT or IOT slots specified must be zero as a result of either placing the item previously contained on some queue or executing a DESTROY statement.

A SELECT statement must be preceded by an EXAMINE statement so that no attempt will be made to select a zero item from a queue. In addition, since the queue definition specifies whether a queue may contain transaction items or IO transaction items,

no attempt must be made to select an available transaction from an I/O transaction queue or an IO transaction from an available transaction queue.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	SELECT	(AT,IOT), q-name

AT

The AT operand specifies that the item selected from the queue is to be placed into the available transaction slot in the current operating CPU.

IOT

The IOT operand specifies that the item selected from the queue is to be placed into the IO transaction item slot in the current operating CPU.

q-name

The "q-name" field specifies the name of the queue for which the select statement is to be executed.

ERROR CONDITIONS

Attempting to SELECT a transaction word into an AT or IOT word which is not zero will cause an error message to be generated and the simulation to terminate.

Attempting to SELECT a zero transaction word from a queue will cause an error message to be generated and the simulation to terminate.

BUFFER

The BUFFER statement provides the ability for the operating system to test the current status of the record and buffer counts for an ordinal file. When a worker routine READ or WRITE statement is executed control is transferred to the operating system by means of a READ/WRITE interrupt. At this point, it is necessary to code a BUFFER statement. The BUFFER statement determines whether a record is available and/or an I/O operation is necessary.

There are no operands required for the BUFFER statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	BUFF	
	NSI	
	NSI+1	
	NSI+2	
	NSI+3	

NSI

The NSI statement is executed if there is a record available and there is no empty buffer for the file specified by the last READ or WRITE statement. The NSI is normally coded as a RETURN to the worker routine which issued the READ or WRITE statement.

NSI+1

The NSI+1 statement is executed if a record is available but a buffer has become empty or full. The fact that a buffer has been depleted indicates an I/O operation should be initiated at that point. Therefore, an I/O transaction item is created for this exit line by the BUFFER statement. Normally, the NSI+1 exit line should transfer control to an IOREADY test. At this point, the I/O operation can either be initiated if all required I/O facilities are available, or else the I/O transaction item may be placed on a queue until the required facilities are available. It should be noted however, that the current available transaction need not be deferred since at least one buffer is currently available.

NSI+2

The NSI+2 statement is executed of there are no records available for the available transaction. This exit line can be reached only if all possible I/O operations have been previously initiated but not yet complete. Since all possible I/O operations have been initiated, no new I/O transaction item is created for this exit line. The normal coding at this point would be to place the worker routine on a deferred queue and to select a new available transaction.

NSI+3

The NSI+3 statement is executed only if the available transaction is an operating system transaction. Since all I/O requests by the operating system are considered to be unblocked and unbuffered each time this exit line is taken an I/O transaction item is created. The normal coding at this point would be a transfer to the IOREADY test.

SEEK

The SEEK statement provides the ability to test an I/O transaction item to determine if a SEEK operation may be performed. In order for the SEEK operation to be performed the I/O transaction item must reference a random access device using a to-from table. If the SEEK statement is used under those conditions the seek time specified in the to-from table will be applied to the device only, and the channel and control unit will be considered free for this time. If a SEEK statement is not used, the seek time specified in the to-from table will be added to the READ/WRITE time and will hold the channel, control unit, and device busy for the entire operation.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	SEEK	

NSI

The NSI statement is executed if a SEEK has been initiated as a result of this statement. If this exit line is taken the IOT is now set to zero. When the seek operation is completed an END OF SEEK interrupt will be generated and the I/O transaction item will be returned to the user.

NSI+1

The NSI+1 statement will be executed if no seek may be performed for the current I/O transaction. If this statement is executed, the I/O transaction item is still available and an IOADVANCE operation will be required.

NOTE

The SEEK statement must not be executed until after an IOREADY has been successfully performed.

ERROR CONDITIONS

Any attempt to execute a RETURN to an available transaction which has no memory allocated to it will generate a diagnostic and terminate the simulation.

IOREADY

The IOREADY statement is used to determine whether the facilities (file, device, and channel control unit pair) required for an IO operation are available. The IOREADY Test may be made on the current IOT or on the current item in a specified queue.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	IOREADY	(q-name)
	NSI	
	NSI+1	

q-name

If the "q-name" field is omitted from this statement, the IOREADY Test is made on the current IOT. If the queue operand is included the IOREADY Test will be performed on the current item in the specified queue. The current item in a queue may be adjusted by means of the EXAMINE statement.

NSI

The NSI statement is executed if the facilities required for this I/O operation are not currently available.

NSI+1

The NSI+1 statement is executed if the facilities required for this I/O operation are available.

Error Conditions

If the q-name operand is included in this statement the queue specified must contain I/O transaction items. Attempting to perform the IOREADY operation on a queue which does not contain I/O transaction items will generate an error message and terminate the simulation.

If the q-name operand is omitted from this statement and the I/O transaction word is zero the simulator will generate an error message and terminate the simulation.

IOADVANCE

The IOADVANCE statement is used to initiate the IO operation for the current IOT. A successful IOREADY statement must have been executed for the current IO transaction item before the IOADVANCE statement may be executed. The interpretation of an IOADVANCE statement will place the current I/O transaction item on the future events chain for the time required to perform the requested IO operation. In addition, the current IOT is set equal to zero.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	IOADV	

There are no operands associated with the IOADVANCE statement.

Error Conditions

If an IOREADY statement has not been executed for the current I/O transaction item when an IOADVANCE

statement is executed, a diagnostic will be issued and the simulation terminated.

Attempting to perform an IOADVANCE statement with a zero I/O transaction word will generate an error message and terminate the simulation.

Once a successful IOREADY statement has been executed for an I/O transaction item, no other IOREADY operation may be performed before the IOADVANCE statement is executed for the same IOT. In general, this means that any compute placed between a successful IOREADY and the corresponding IOADVANCE must have all interrupts disabled.

IOTERM

The IOTERM statement is used to select transaction items off of the deferred queue once the required I/O operation has been performed. A transaction item is normally placed on the I/O deferred queue because it has requested a record which is not currently available. In order for there to be no records available for a given file the buffer for that file must have been previously emptied. Every time a buffer is emptied the BUFF statement creates an I/O transaction item which will be used to fill the buffer again. Therefore, once the I/O operation requested by the I/O transaction item has been completed the transaction item which was placed on the I/O deferred queue may once again proceed.

The IOTERM statement tests the item indicated by the current setting of the queue pointer, for the queue specified, to determine if the item is a transaction item which can now proceed due to the termination

of the current I/O transaction item. A transaction item is considered to be able to proceed if it generated the current I/O transaction item, and if the current I/O transaction item has filled the empty buffer which caused this I/O transaction item to be placed on the I/O deferred queue. A transaction item may not be selected off the I/O deferred queue if any opened file has one or more empty buffers.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	IOTERM	q-name
	NSI	
	NSI+1	

q-name

The "q-name" field specifies the name of the queue which is to be examined by the IOTERM statement.

NSI

The NSI statement will be executed if the current item in the specified queue may not be selected.

NSI+1

The NSI+1 statement will be executed if the current item in the queue specified may be selected and removed from the I/O deferred queue.

Error Conditions

If the queue specified by the q-name operand does not contain transaction items a diagnostic message will be supplied and the simulation terminated.

Operating System Switches

There are three statements provided in the simulator which handle the setting and testing of switches provided for use by operating systems and primary worker routines. The three statements are SET, RESET, and TEST. The SET statement may be used to turn one of the available switches on, the RESET statement may be used to turn one of the switches off, and the TEST statement may be used to examine one of the available switches and determine whether it is currently on or off.

Local Switches

The simulator provides twenty switches which are considered local to a single CPU. These local switches are numbered consecutively from one to twenty. Local switches are particularly useful when a single operating system is shared between two or more CPU's. For example, if CPU number one is currently executing in the operating system and

executes a set 1 instruction switch number one for CPU number one will be turned on. However, switch number one for CPU number one is distinct from switch number one for CPU number two. Therefore, if CPU number two were now to execute a test 1 instruction that switch might well be off.

Global Switches

The simulator provides one hundred global switches numbered consecutively from 101 to 200. The global switches provided may be manipulated by the three statements above in the same way as local switches. However, these switches may be referenced by any one of the CPU's in a single simulation. Specifically, if CPU number one references switch 101 and CPU number two references switch number 101 they are both referencing the same switch.

SET

The SET statement is used to turn on one of the local or global switches provided by the simulator. If the switch to be SET is already on, it will be left on.

STATEMENT FORMAT		
LABEL	OP-CD	OPERAND
	SET	switch

switch

The "switch" field specifies the number of the switch to be turned on. If the number of the switch is in the range from 1 to 20 the switch turned on will be a local switch. If the number of the switch to be turned on is in the range from 101 to 200 the switch turned on will be a global switch.

RESET

The RESET statement is used to turn off one of the local or global switches provided by the simulator. If the switch to be RESET is already off it will be left off.

STATEMENT FORMAT		
LABEL	OP-CD	O-ERANDS
	RESET	switch

switch

The "switch" field specifies the number of the switch to be turned off. If the number of the switch is in the range from 1 to 20 the switch to be turned off is a local switch. If the number of the switch to be turned off is in the range from 101 to 200 the switch to be turned off is a global switch.

TEST

The TEST statement allows the user to determine if one of the local or global switches provided by the simulator is currently on or off.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	TEST	switch
	NSI	
	NSI+1	

switch

The "switch" field specifies the number of the switch to be tested. If the number of the switch is in the range from 1 to 20 the switch to be tested is a local switch. If the number of the switch is in the range from 101 to 200 the switch to be tested is a global switch.

NSI

The NSI statement will be executed if the switch tested is currently on.

NSI+1

The NSI+1 statement will be executed if the switch to be tested is currently off.

NOTE

At the beginning of any simulation all of the available switches are initialized to the off position.

INTERRUPT

The INTERRUPT statement allows an operating system to generate an interrupt which will be placed at the top of the future events chain. However, the current operating program will not lose control of the CPU until some statement which advances simulated time is executed. If the user wishes the interrupt to take place immediately the interrupt statement may be followed by a COMPUTE statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	INTER	cpu-name, int-name

cpu-name

The "cpu-name" field specifies the name of the CPU which is to receive this interrupt.

int-name

The int-name operand specifies the name of the interrupt which will be generated for the specified CPU.

It should be noted that specifying the name of an interrupt which has previously been defined by the system may be a cause of problems. For example, if the interrupt name specified is normally the CLOSE interrupt the operating system may expect to find an I/O transaction item where in fact the interrupt generated by this statement will not provide an I/O transaction item.

ERROR CONDITIONS

The specification of an interrupt name which has not been defined for the CPU specified will cause the generation of an error message and the termination of the simulation.

DISABLE

The DISABL statement allows the user to defer the occurrence of any or all interrupts. If an interrupt generating transaction, upon reaching the top of the future events chain, finds that its interrupt has been disabled, it will remain on the future events chain until an ENABLE statement specifying its interrupt allows it to enter the interrupt routine.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	DISABL	(All, int-name)

ALL

The ALL operand specifies that all interrupts defined for the current CPU are to be disabled.

int-name

The "int-name" field specifies the name of a single interrupt which is to be disabled for the current CPU.

ERROR CONDITIONS

Specifying the name of an interrupt which has not been defined for the current CPU by means of the int-name operand will cause the generation of an error message and the termination of the simulation.

ENABLE

The ENABLE statement allows the user to reverse the action specified in a DISABL statement. If there is a transaction item at the top of the future events chain which could not generate an interrupt because its interrupt was disabled, the interrupt will occur as soon as a CPU advance time occurs.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	ENABLE	(ALL, int-name)

ALL

The ALL operand specifies that all disabled interrupts are now to be enabled.

int-name

The "int-name" field specifies the name of the single interrupt which is to be enabled by this statement.

ERROR CONDITIONS

If the name of an interrupt which has not been defined for the current CPU is specified by means of the int-name operand a diagnostic message will be issued and the simulation terminated.

CLOCK

There is one simulated interval timer for each CPU in a given simulation. When the interval timer for any given CPU reaches zero a clock interrupt for that CPU will occur unless the CLOCK interrupt is disabled. Therefore, the CLOCK statement is always interpreted relative to the current operating CPU.

The CLOCK statement is used to set the simulated interval timer for the current operating CPU. When the CLOCK statement is interpreted, the value of the TIME operand is placed into the simulated interval timer destroying any previous setting. Once the interval timer has reached zero, thereby generating an interrupt, no new interrupt will occur until the interval timer has been reset by a new CLOCK statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	CLOCK	time

time

The "time" field specifies the time in microseconds which is to be placed into the interval timer for the current operating CPU. The value of this field must be in the form DDDE. The format DDDE is interpreted as a number in the range from 1 to 999 with an exponent in the range from 0 to 9.

RETURN

The RETURN statement is used to leave the operating system and return to the next sequential instruction for the available transaction. The available transaction will normally represent a worker routine, but it may also represent a primary worker or another operating system transaction. When a RETURN statement is interpreted, the current operating transaction (COT), representing the operating system, is destroyed and the available transaction (AT), representing the worker routine, becomes the current operating transaction.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	RETURN	

There are no operands associated with the RETURN statement.

Error Conditions

A RETURN statement may not be executed if there is no available transaction or if there is an I/O transaction in the current CPU item. If either of these error conditions are found to exist an appropriate diagnostic message is generated and the simulation is terminated.

ACTIVATE

The ACTIVATE statement causes the immediate generation of a transaction for the worker routine specified by the job-name operand. In addition, the transaction word for the newly generated transaction is placed into the available transaction (AT) position in the current CPU item. Before the operating system executes an ACTIVATE statement, the available transaction in the current CPU item must be set to zero by using a DESTROY statement or a PLACE statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	ACTIV	job-name

job-name

The "job-name" field must contain the name of the worker routine for which a transaction is to be generated by the ACTIVATE statement.

Error Conditions

Attempting to execute an ACTIVATE statement when the available transaction word is not zero in the current CPU item will cause the generation of a diagnostic message and the termination of the simulation.

Attempting to activate a transaction for a worker routine which has not been defined for the current simulation will cause the generation of an error diagnostic and the termination of the simulation.

RECEIVE

The RECEIVE statement takes the transaction word associated with a transaction which has just generated a RECEIVE interrupt and makes it the available transaction for the current operating CPU. Before executing the RECEIVE statement, the operating system must insure that the available transaction word for the current CPU is zero. This may be performed by either the PLACE statement or the DESTROY statement.

In a multiprocessor system where two or more CPU's share the same operating system, the operating system must insure that the CPU which accepted the RECEIVE interrupt is the same CPU which executes the RECEIVE statement. Therefore, any statements placed between the beginning of the RECEIVE interrupt vector and the RECEIVE statement which advances time, must have all interrupts disabled.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	RECEIV	

There are no operands associated with the RECEIVE statement.

Error Conditions

If the RECEIVE statement is executed by any CPU which does not have a receiver interrupt pending a diagnostic message will be issued and simulation terminated.

CYCLE

The CYCLE statement allows an operating system to enter the WAIT state when the CPU has no further processing to do. Normally, the CYCLE statement will be issued only when there is no available transaction in the entire system. Before executing a CYCLE statement the operating system must insure that all interrupts are enabled, the available transaction word is zero, and the I/O transaction word is zero.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	CYCLE	

There are no operands associated with the CYCLE statement.

Error Conditions

If the CYCLE statement is executed when the available transaction or the I/O transaction for the

current CPU item is not zero, a diagnostic message will be issued and the simulation terminated.

If a CYCLE statement is executed when there is no possible interrupt on the future events chain an error diagnostic will be issued and the simulation terminated.

DESTROY

The DESTROY statement causes the specified transaction word for the current CPU item to be cleared to zero and the associated transaction item to be removed from the system.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	DESTROY	(AT,IOT)

AT

The AT operand specifies that the available transaction word for the current CPU item is to be set to zero and the available transaction item is to be cleared from the system. The AT operand is normally used only as the result of a worker program termination.

IOT

The IOT operand specifies that the IO transaction word for the current CPU item is to be set to zero and the associated IO transaction item is to be

cleared from the system. The IOT operar. is normally used only as a result of the termination of an I/O operation.

Error Conditions

If the CPU item specified by the DESTROY statement is currently zero a diagnostic message will be issued and the simulation terminated.

Any attempt to destroy an available transaction item (AT) which has not closed all opened files will cause a diagnostic message to be generated and the simulation to terminate.

PERIPHERAL

The PERIPHERAL statement allows the files and devices required by a transaction item to be examined and, if available, assigned to that transaction item. The PERIPHERAL statement uses the information provided by the user in the OF statement supplied for every worker routine. Only those files and devices which the user specified must be seized by a transaction will be assigned by the PERIPHERAL statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	PERIPH	q-name

q-name

If the "q-name" field is omitted from this statement the PERIPHERAL operation will be performed with respect to the available transaction for the current CPU item.

If the QUEUE operand is supplied for this statement the PERIPHERAL operation will be executed

with respect to the current transaction item in the specified queue. The current transaction for a queue may be selected by means of the EXAMINE statement.

Error Conditions

If the "q-name" field is specified, the queue must not contain I/O transaction words. If a queue containing I/O transaction words is supplied a diagnostic message will be issued and the simulation terminated.

DCYCLE

The DCYCLE statement allows an operating system to place the current CPU into the WAIT state until a specified interrupt occurs. Since all interrupts except the one specified will be disabled the operating system must insure that there will be at least one interrupt generated from the specified interrupt source.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	DCYCLE	int-name

int-name

The "int-name" field specifies the name of the interrupt which is to be left enabled when the DCYCLE statement is interpreted. All interrupts other than the one specified in the "int-name" field will be disabled.

ERROR CONDITIONS

If there are not interrupts on the future events chian for the single interrupt left enabled a diagnostic message will be issued and the simulation terminated.

MATCH

The MATCH statement is used to test whether the current transaction item on the specified queue has been delayed until the termination of the available transaction. In this respect, it operates much like the IOTERM statement does for I/O transactions. The MATCH statement would normally be coded in the program termination interrupt vector. A transaction item would normally be placed on the MATCH queue due to a CALL statement with a DELAY operand.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MATCH	q-name
	NSI	
	NSI+1	

q-name

The "q-name" field specifies the name of the queue for which the MATCH statement is to be executed.

NSI

The NSI statement will be executed if a MATCH condition is found to exist between an item on the queue specified and the available transaction.

NSI+1

The NSI+1 statement will be executed if NOMATCH is found to exist between the available transaction and any items on the specified queue.

Error Conditions

If the available transaction word for the current CPU item is zero when a MATCH statement is executed a diagnostic message will be issued and the simulation terminated.

If the QUEUE operand specifies the name of a queue which contains I/O transaction items or a queue where the current entry is zero, a diagnostic message will be issued and the simulation terminated.

SELECT/PLACE

The SELECT/PLACE statement allows the user to move a transaction word from one queue to another without altering the current CPU item in any way.

STATEMENT FORMAT		
LABEL	OP CD	OPERANDS
	SELPLC	q-name 1, q-name 2 (,AT,IOT)

q-name 1

The "q-name 1" field specifies the name of the queue from which the AT or IOT will be selected.

q-name 2

The "q-name 2" field specifies the name of the queue onto which the AT or IOT selected from QUEUE1 will be placed.

AT

The AT operand specifies that both "q-name 1" and "q-name 2" contain a transaction item.

IOT

The IOT operand specifies that both "q-name 1" and "q-name 2" contain I/O transaction items.

Error Conditions

If either or both of the two queues specified in this statement do not contain the type of transaction specified by the AT or IOT operands an error message will be issued and the simulation terminated.

END

The END statement must be the last statement of every routine. The END statement is not executable but is used to signify the end of every assembly. The statement used to terminate execution of a routine is the TERM statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	END	

RECEIVING CPU

The RCV statement is used to specify the name of the CPU which is to receive the current W/R if it is other than CPU number one. In addition, the RCV statement may be used to specify the number of the load class entry to be used by the W/R if it is other than number one. This statement must come after a JOB statement and before an END statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	RCV	receiving-cpu-name, load-class#

receiving-cpu-name

The "receiving-cpu-name" must be the name of the CPU which is to receive this worker routine when it is introduced to the system.

load-class-#

The "load-class-#" field must contain an integer number from 1 to 15 which specifies the load class entry to be used by the W/R.

OPERATING SYSTEM

The OS statement must be included in each operating system described for a simulation run. The statement describes the memory module that the operating system is to occupy as well as the CPUs it is to control.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	OS	mem-name, cpu-name (,cpu-name)

mem-name

The "mem-name" field must contain the name of the memory module the operating system is to occupy.

cpu-name

The "cpu-name" field must contain the name of the CPU which the operating system is to control.

ASSEMBLY

The ASSEMBLY statement is used to delimit the system parameters from the worker routines. In addition, it provides the user with control over the printing of the assembly output. This statement must immediately preceed the first JOB statement in the input stream.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	ASSEMBLY	(PRINT, NOPRINT)

PRINT

If the PRINT option is specified the worker routine object code will be printed after the diagnostic messages.

NOPRINT

If the NOPRINT option is specified the worker routine object code will not be printed.

If the operand is given, the PRINT option will be assumed.

INTERRUPT VECTOR

The IV statement is used to change the label which specifies the starting location in an operating system for interrupt processing. Normally the user will label the statement which is to receive control of the CPU when a specified interrupt occurs with an interrupt name as shown below. However, if the user wants to use a name which is different from the one specified, he may do so by use of the IV statement.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	IV	old-int-name,new-int-name,cpu-name (,cpu-name)

old-int-name

The "old-int-name" field must contain the name by which the interrupt to be changed is currently known to the system.

new-int-name

The "new-int-name" field must contain the new name, limited to six characters, by which this interrupt is to be known.

cpu-name

The "cpu-name" field must contain the name or names of the CPUs for which this change is to be effected.

Interrupt Vector Table

The first ten interrupts in the system are fixed for specific interrupt conditions and must be referred to by the following standard names:

Interrupt Name	Interrupt Condition
1. IOTINT	I/O Termination
2. RWINT	Read or Write Operation
3. FNCINT	Function Operation
4. CLKINT	Clock Interrupt
5. RCVINT	Receive Interrupt
6. TRMINT	Program Termination
7. EDSINT	End of Seek Operation
8. OCINT	Open or Close Operation
9. CDLINT	Call with Delay Option
10. CALINT	Call with No Delay Option

Interrupts 11 through 20 may be assigned functional significance by the user but must be referred to by the following names:

11. INT11

12. INT12

13. INT13

14. INT14

15. INT15

16. INT16

17. INT17

18. INT18

19. INT19

20. INT20

APPENDIX A

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

MULTIPLE-CARD LAYOUT FORM

Form X24-6599-0
Printed in U. S. A.

Company ______ Application ______ by ______ Date ______ Job No. S-3 Sheet No. ______

Card 01

Columns	Field
1–2	CARD CODE (01)
3–6	CPU ID
7–10	LOGICAL DATA UNIT
11–12	DECIMAL ARITHMETIC TIMES: DEC DIGIT
13–19	DECIMAL ARITHMETIC TIMES: ADD TIME (µ SEC)
20–26	DECIMAL ARITHMETIC TIMES: MULTIPLY TIME (µ SEC)
27–33	DECIMAL ARITHMETIC TIMES: DIVIDE TIME (µ SEC)
34–40	FIXED POINT ARITHMETIC TIMES: ADD TIME (µ SEC)
41–47	FIXED POINT ARITHMETIC TIMES: MULTIPLY TIME (µ SEC)
48–54	FIXED POINT ARITHMETIC TIMES: DIVIDE TIME (µ SEC)
55–61	FLOATING POINT ARITHMETIC TIMES: ADD TIME (µ SEC)
62–68	FLOATING POINT ARITHMETIC TIMES: MULTIPLY TIME (µ SEC)
69–75	FLOATING POINT ARITHMETIC TIMES: DIVIDE TIME (µ SEC)

Card 02

Columns	Field
1–2	CARD CODE (02)
3–6	CPU ID
7–8	LINE INST
9–10	DEC DIG
11–12	CHAR
13–19	MOVE TIME
20–26	MOVE & EDIT TIME
27–30	FIXED POINT TABLE: E 1
31–34	FIXED POINT TABLE: E 2
35–38	FIXED POINT TABLE: E 3
39–42	FIXED POINT TABLE: E 4
43–46	FLOATING POINT TABLE: E 1
47–50	FLOATING POINT TABLE: E 2
51–54	FLOATING POINT TABLE: E 3
55–62	GIBSON MIX FACTOR
63–69	ADDRESS ADJUST FACTOR

Card 03

Columns	Field
1–2	CARD CODE (03)
3–6	MEM ID
7–10	MEM ACCESS UNIT
11–17	MEM CYCLE TIME
18–24	MEM ACCESS TIME
25–33	MEM SIZE
34–39	PAGE SIZE

Card 04

Columns	Field
1–2	CARD CODE (04)
3–6	CHAN ID
7	TYPE
8–14	SEL OR BURST RATE
15–20	% CPU INT
21–27	MPX RATE
28–33	% CPU INT

Card 05

Columns	Field
1–2	CARD CODE (05)
3–6	DEV ID
7	TYPE
8–15	DEV RATE
16–19	TRAN WIDTH
20–26	START TIME
27–33	STOP TIME
34–36	RWND TIME
37–43	TIME BEFORE PENALTY
44–50	PENALTY TIME
51–57	FORM TIME

WORKING PAPER

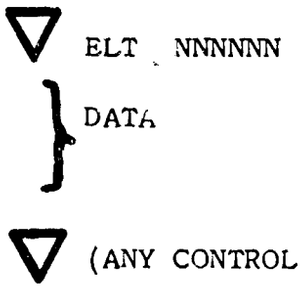
APPENDIX B

WORKING PAPER

Entering Data Into the Library

Facilities are provided within the S-3 Assembler for placing Macro Definitions, Hardware Definitions, and Configuration Data into the library. These facilities have been provided because these items must be formatted by the Assembler before they are placed in the library, and no attempt should be made to enter these items into the library without using the Assembler.

Anything else which the user might wish to have placed into the system library, such as worker routines, operating systems, etc., may be entered as follows:



Where ▽ is a 7/8 punch and NNNNNN is the 6 character name of the item to be entered.

This data may then be retrieved by means of the COPY statement as described in the library facility.

WORKING PAPER

APPENDIX C

WORKING PAPER

Poisson Distribution

The Poisson Distribution is used to vary the values of the generation interval about the specified average value. That is, when the POISS option is specified in a GENERATE statement, the statement does not cause a transaction to be generated at the interval specified. Rather, the interval is used as an average value, and the intervals between specific generations is varied in such a manner as to preserve the specified value as the average.

The Poisson Distribution is the analogue, for discrete events, of the well known normal distribution (bell-shaped curve). The Poisson table is used as follows. When a GENERATE statement generates a transaction, S-3 then computes when the next transaction is to be generated by that statement and places it on the FEC. Using the POISS option, this computation is performed by first generating a random number between 0 and 99. This random number is then used as the

argument for a table look-up in the Poisson table. This table actually contains values not for every integer from 0 to 99, but for every 5. That is, there are twenty values in the table, one for the range of 0-4, one for the range of 5-9, and so on through the range of 95-99. The number found in the Poisson table is then multiplied by the interval given in the GENERATE statement, to obtain the next generation interval.

Altering the S-3 Poisson Distribution Table

The Poisson Distribution Table in S-3 is initialized by Subroutine ZERO. The table is located in AT20(81) to AT20(100) in Subroutine ZERO and consists of twenty words.

Since a random draw is made from one of these twenty values, the sum of the twenty values in the table for a normal distribution must be equal to 20 in order to preserve the average of the table. To create a flat or null distribution, all twenty entries in the table may be initialized to a value of one.

WORKING PAPER

APPENDIX D

WORKING PAPER

S-3 Assembler Statement Summary

PART I: Hardware Definitions and Configuration Data

STATEMENT FORMAT

LABEL	OP-CD	OPERANDS
	CPU-DEF	cpu-id (,LIB,CAT,NCAT)
	CPU-END	
	MEM-DEF	mem-id (,LIB,CAT,NCAT)
	MEM-END	
	CHAN-DEF	chan-id (,LIB,CAT,NCAT)
	CHAN-END	
	DEV-DEF	dev-name (,LIB,CAT,NCAT)
	DEV-END	
	CONFIG	(config-name) (,LIB,CAT)
	CPU	cpu-name, cpu-id
	MEM	mem-name, mem-id, cpu-name . . .
	CHANNEL	chan-name, chan-id, cpu-name . . .
	CONTROL	(IN,OUT,I/O), ctl-name, chan-name . . .
	DEVICE	(SEIZE,NOSEIZE), dev-name, dev-id, ctl-name . . .
	CONFIG-END	

PART II: System Control Parameters

STATEMENT FORMAT

LABEL	OP-CD	OPERANDS
	TF-DEF	dev-name #-files
	TABLE	N11, N12, N13
	TF-END	
	Q-DEF	
	QUEUE	q-name, entries (,AT,IOT) (,FIFO,LIFO,PRI)
	Q-END	
	FILE-DEF	
	RF	file-name, dev-name, rel-loc, buff-len, rec/buff, buff/file
	RFC	old-file-name, new-file-name (,dev-name) (,rel-loc)
	FILE-END	
	LC-DEF	
	LOAD	class-#, cpu-name (,cpu-name)
	LC-END	
	RC-DEF	
	RUN	loading-cpu-name, cpu-name (,cpu-name)
	RC-END	
	TAB-CTL	(ON,OFF) (,T1) (,T2) (, . . .)
	STAT-CTL	(ON,OFF) (,ST1) (,ST2) (, . . .)
	STATISTICS	minutes, count
symbol	GEQU	value
symbol	LEQU	value

PART III: S-3 Processing Statements

STATEMENT FORMAT

LABEL	OP-CD	OPERANDS
	ACTIV	job-name
	ALLOC	mem-name 1, mem-name 2
	ASSEMBLY	(PRINT,NOPRINT)
	BUFF	
	CALL	job-name, (DLY,NODLY)
	CLOCK	time
	CLOSE	of-name, (REWIND,NOREWIND)
	COMPUT	inst, (mark)
	CYCLE	
	DALLOC	
	DCYCLE	int-name
	DESTROY	(AT,IOT)
	DISABL	(ALL, int-name)
	ENABLE	(ALL, int-name)
	EOF	of-name, loc
	EXAM-F	q-name
	EXAM-L	q-name
	EXAM-N	q-name
	EXEC	int-name
	EXIT	
	FUNCTN	fno
	GEN	interval, (FIX,POISS), limit, priority
	INTER	cpu-name, int-name
	IOADV	
	IOREADY	(q-name)
	IOTERM	q-name
	JOB	job-name, (OS,PW,BUS,SCI), (REENT,NREENT)
	LOOP	count, loc
	MATCH	q-name
	MEMORY	mem-name 1, mem-name 2 (,q-name)
	MEM-1	inst, char
	MEM-2	dec-field, dec-len, bin-fields, bin-len, float-fields, float-len

PART III: S-3 Processing Statements

STATEMENT FORMAT

LABEL	OP-CD	OPERANDS
	MOVE	char
	MOVE-E	char
	OF	of-name, rf-name, buff-no, (NOSEIZE,SEIZE-DEV, SEIZE-FILE)
	OPEN	of-name, (INPUT,OUTPUT)
	OS	mem-name, cpu-name (,cpu-name)
	PACK	mem-name 1, mem-name 2
	PERIPH	q-name
	PLACE	(AT,IOT), q-name
	PRINT	of-name, spaces
	RCV	receiving-cpu-name, load-class#
	READ	of-name
	RECEIV	
	RESET	switch
	RETURN	
	SEEK	
	SELECT	(AT,IOT), q-name
	SELPLC	q-name 1, q-name 2 (,AT,IOT)
	SET	switch
	SUB	loc
	SWITCHES	
	TERM	(job-name)
	TEST	switch
	TRA	loc
	TRA-P	PERCENT,LOC
	WRITE	of-name

WORKING PAPER

SECTION II

USACSSEC S-3

MACRO AND LIBRARY SPECIFICATIONS

Presented by
Leo J. Cohen Associates, Inc.

October 1, 1968

Revised October 24, 1968

334 West State Street
Trenton, New Jersey
08618
(609) 695-1488

110 North Royal Street
Suite 305
Alexandria, Virginia
22314
(703) 548-0128

WORKING PAPER

Table of Contents

MACRO DEFINITIONS

The S-3 Assembler may be used to define macros and place them into the library. However, a macro may not be used in the same run it is defined in. A macro definition must be enclosed between a MDEF and an MEND statement. The format used for defining macros is shown in the following figure.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MDEF	
(&sym)	macro-name	(&arg)(,&arg...)
(.stmt-var.)(&sym,sym,&arg)(*)	(&arg,op-cd,macro-name)	(&arg(*),operand(*))
	.	
	.	
* comments		
	MEND	

The first line of code above shows the proper use of the MDEF statement.

The second line above shows the format for a Macro Prototype statement. The prototype statement is an exact replica of the statement which will be used to

invoke the use of this macro during the course of an assembly. A detailed description of the fields which may be used is provided below.

& sym

The '& sym' field as shown above may be replaced by a macro symbol of up to twelve characters. A macro symbol, or symbol variable, must begin with an ampersand followed by an alphabetic character followed by any alphameric character. If the '& sym' field is coded, the label provided with a macro call, if any, may be substituted into the macro expansion.

macro-name

The 'macro-name' field must contain the name (up to six alphameric characters) by which this macro will be referenced during the course of an assembly. Care must be taken to ensure that the 'macro-name' provided does not duplicate any entry in the table of contents for the Program Complex File used during assembly.

& arg

The '& arg' operand may be used to pass parameters to the macro during an assembly. The operands used when a macro is invoked replace every occurrence of the symbolic argument coded in the macro definition.

The third line of code above shows the format for a macro statement. A detailed description of the possible fields follows:

.stmt-var.

If the '.stmt-var.' field is used 'stmt-var' must be replaced with any two alphabetic characters. The generation of this statement may then be controlled by means of the MSET statement as shown later on.

& sym

If the user wants to have a statement in the macro expansion labeled the same as the label supplied with the macro call, the statement in the macro definition must have the same symbolic variable as the prototype statement.

sym

If the user wants to have a statement in the macro expansion labeled for reference elsewhere in the macro expansion a symbol may be used to label the statement. Note that care must be taken to ensure that the symbol is not duplicated in the rest of the assembly, and that the same macro may not be invoked more than once unless the '*' operand is included for all labels.

& arg

If the user wants to be able to provide a label for a statement when the macro is invoked then the symbolic variable used to describe that operand in the prototype statement should be coded as the label for that statement.

*

The '*' symbol may be appended to any of the permissable labels of a macro statement if the user wants to insure that all labels in the

program are unique. If the '*' symbol is used a two digit macro level number is appended to the symbol provided. Therefore, if the '*' symbol is used, the maximum length of the symbol is 10 alphameric characters. Caution must be taken to insure that if the '*' symbol is used, every reference to the symbol it is appended to, also has the *. Otherwise, those symbols will be unresolved.

& arg

If the user wants to pass an operation code to the macro expansion when the macro is invoked, the symbolic argument coded in the prototype statement may be coded as the operation code of a macro statement. When the macro is invoked, the operation code supplied as an operand to the macro will replace the symbolic argument used as an operation code in the macro statement.

op-cd

If the operation code to be used in a macro statement is known when the macro is defined then it may be coded in the 'op-cd' field.

macro-name

The user may invoke other macros within the definition of a new macro. Macros may be nested to any level, and may have a maximum of 50 operands.

& arg

If the argument for a macro statement is to be supplied when the macro is expanded, the '& arg' field from the prototype statement should be coded in the macro statement. When the macro is invoked, the argument supplied in the macro call will be used to substitute for the symbolic argument.

operand

If the operand for a macro statement is known at the time the macro is defined it may be coded as a normal operand, rather than as a symbolic operand.

*

If the user wants to reference a statement label which makes use of the macro level counter, then the '*' operand should be appended to that label for every reference.

Lines 4 and 5 are used to show that the type of statement shown in line 3 may be repeated as often as necessary. The assembler is capable of handling approximately 400 macro statements for a single macro definition.

Line 6 demonstrates that comments may be included within a macro definition. However, all comments within a macro definition must

be on comments cards. They may not be placed on the same line as macro statement.

Line 7 shows the proper use of the MEND statement.

MACRO SET

The MACRO SET statement is used to control the generation of statements within a macro definition. A detailed description of the statement is provided below.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MSET	stmt-var,value[,stmt-var,value,...]

stmt-var

The 'stmt-var' field must contain a two character alphabetic symbol as a 'stmt-var' in a macro description.

value

The 'value' field specifies the maximum number of statements with the 'stmt-var' field supplied above which are to be generated in the next macro expansion.

MACRO EXAMPLES

MACRO DEFINITION 1

```
          MDEF
& SYM     SEARCH    &A,&B,&C,&D
& SYM     EXAM-F    &A
          TRA       &C
TEST *    &B        &A
          TRA       &D
          EXAM-N    &A
          TRA       &C
          TRA       TEST *
          MEND
```

MACRO CALL 1A

```
          END-IO    SEARCH    DQ,IOTERM,EXIT1,EXIT2
G         END-IO    EXAM-F    DQ
G                   TRA       EXIT1
G         TEST01    IOTERM    DQ
G                   TRA       EXIT2
G                   EXAM-N    DQ
G                   TRA       EXIT1
G                   TRA       TEST01
```

END OF MACRO

MACRO CALL 1B

```
            SEARCH      IQ,IOREADY,EXIT1,EXIT2
G           EXAM-F      IQ
G           TRA         EXIT1
GTEST02     IOREADY     IQ
G           TRA         EXIT2
G           EXAM-N      IQ
G           TRA         EXIT1
G           TRA         TEST02
END OF MACRO
```

MACRO DEFINITION 2

```
                        MDEF
            INOUT       &A1,&A2,&A3,&A4,&B1,&B2,&B3,&B4
.AA.        OF          FILE1,&A1,1,NOSEIZE
.AA.        OF          FILE2,&A2,1,NOSEIZE
.AA.        OF          FILE3,A3,1,NOSEIZE
.AA.        OF          FILE4,A4,1,NOSEIZE
.BB.        OF          FILE5,B1,1,NOSEIZE
.BB.        OF          FILE6,B2,1,NOSEIZE
.BB.        OF          FILE7,B3,1,NOSEIZE
.BB.        OF          FILE8,B4,1,NOSEIZE
            MEND
```

MACRO CALL 2A

```
          MSET      AA,1,BB,1
          INOUT     INPUT,,,,OUTPUT
G         OF        FILE1,INPUT,1,NOSEIZE
G         OF        FILE5,OUTPUT,1,NOSEIZE
END OF MACRO
```

MACRO CALL 2B

```
          MSET      AA,2,BB,2
          INOUT     INPUTA,INPUTB,,,OUTPUTA,OUTPUTB
G         OF        FILE1,INPUTA,1,NOSEIZE
G         OF        FILE2,INPUTB,1,NOSEIZE
G         OF        FILE5,OUTPUTA,1,NOSEIZE
G         OF        FILE6,OUTPUTB,1,NOSEIZE
END OF MACRO
```

COPY

The COPY statement is used to take information from the system library and insert it into the input data stream at the current location. A COPY statement may not bring another COPY statement into the input stream and a COPY statement may not be included in a macro definition. However, a COPY statement may bring macro calls or macro definitions into the input stream.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	COPY	data-name

data-name

The "data-name" field must contain the name (limited to 6 characters) by which the data to be brought into the input stream is known.

MACRO EQUATE

The MEQU statement is provided for use within macro definitions to supply a limited capability for calculation.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	MEQU	A, B, C

symbol

The symbol supplied in this field will be entered into the local symbol table for the worker routine in which this macro is used.

A, B, C

The value assigned to the symbol above will be calculated as follows:

value = (A * B) / C

Any or all of the operands a, b, c may be substituted for by the use of macro variables.

IF

The IF statement provides the macro generator with a limited logical capability.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	IF	stmt-var1,val-1,stmt-var2,val-2(,...)

stmt-var1

The field "stmt-var1" must contain a 2 character statement variable.

val-1

If the "stmt-var1" has a current value of "val-1" then the other (stmt-var, val) operand pairs will be treated as if they were operands of a MSET statement. If the current value of "stmt-var1" is not equal to "val-1" no MSET operation will take place.

STORE

The STORE statement provides the user with the ability to place input to the assembler into the library. When a STORE statement is interpreted, all data from the input stream is placed onto the CUR output file until an END-STORE statement is encountered.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
	STORE	file-name
Statements to be placed in library go here.		
	END-STORE	

file-name

The "file-name" field must contain the name by which this data will be known in the library. This field is limited to six characters. The first of which must be alphabetic.

NOTE:

Between a STORE and an END-STORE no statement with a CAT option is accepted.

UPDATE

The UPDATE macro provides the user with the ability to generate a simple update program model with a minimum of effort.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
(sym)	UPDATE	input-of-name,output-of-name,trans-of-
		name,prob,inst,records

input-of-name

The "input-of-name" field must contain the name of the input master file as specified in the ordinal file statement.

output-of-name

The "output-of-name" field must contain the name of the output master file as specified in the ordinal file statement.

trans-of-name

The "trans-of-name" field must contain the name of the input transaction file as specified in the ordinal file statement.

prob

The "prob" field must contain the probability that any one record on the input master file will have a transaction to be processed against it. This probability must be expressed as a number from 1-99.

inst

The "inst" field must contain the number of instructions required to process one transaction.

records

The "records" field must contain the number of records on the input master file.

EXAMPLE

Suppose the user wants to generate a model of an update program which has a 10,000 record master file, with 5% transaction and 1000 instructions required for each transaction. This example would be coded as follows:

```
JOB          UPDT1,BUS,NRBENT
OF           MASTIN,RF1,4,SBIZE-FILE
OF           MASTOUT,RF2,4,SBIZE-FILE
OF           TRANS,RF3,2,SBIZE-FILE
```

```
MEM-1           23,13

GEN             10,FIX,1,10

UPDATE          MASTIN,MASTOUT,TRANS,5,1000,10000
```

SORT

The SORT macro is used to generate a model of a typical sort package. The user is required to provide ordinal file statements for the input file, work files, and the output file. The SORT macro is currently capable of handling a balanced sort for four or six work files.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
(sym)	SORT	input-of-name,output-of-name,#-work-files,
		records, merges, mem-records

input-of-name

The "input-of-name" field must contain the name of the input file as specified in the ordinal file statement.

output-of-name

The "output-of-name" field must contain the name of the output file as specified in the ordinal file statement.

#-work-files

The "#-work-files" field must contain the number of work files available to this sort. Work files must be named WORK01, WORK02, etc. The number of work files specified must equal four or six.

Records

The "records" file must specify the number of records contained in the input file. The number of records in the input file should be an even multiple of the number of records which can be contained in memory as specified in the "mem-records" field.

merges

The "merges" field must contain the number of merge passes required by the sort.

mem-records

The "mem-records" field must contain the number of records which can be entered into memory at one time. This is dependent on the amount of data storage available.

EXAMPLE

Suppose the user wants to sort 10,000 fifty character records using four work files. For this example we will assume that the file has a certain degree of order and that four merge passes will be enough. The example would be coded as follows:

```
JOB         SORT1,BUS,NREENT
OF          INPUT,RF1,2,NOSEIZE
OF          OUTPUT,RF2,2,SEIZE-FILE
OF          WORK01,RF3,2,SEIZE-DEV
OF          WORK02,RF4,2,SEIZE-DEV
OF          WORK03,RF5,2,SEIZE-DEV
OF          WORK04,RF6,2,SEIZE-DEV
MEM-1       53,53
GEN         1,FIX,1,10
SORT        INPUT,OUTPUT,4,10000,4,100
```

Notice that no END statement is required for this job since it will be generated by the SORT macro.

MERGE

The MERGE macro is used to generate a model of a typical merge package. The user is required to provide ordinal file statements for the files to be merged as well as the output file. The MERGE macro is currently capable of handling from two to six input files.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
(sym)	MERGE	no-input,records,output-of-name

no-input

The "no-input" field must contain the total number of input files to be merged. This field must contain a number from two to six. The names of the input files as specifies in ordinal file statements must be INPUT01,INPUT02, etc.

records

The "records" field must contain the total number of records on all of the input files.

output-of-name

The "output-of-name" field must contain the ordinal file name for the file on which the merged output is to be written.

EXAMPLE

Suppose the user wants to generate a model to represent the merging of three files onto a single file. The three input files contain about 10,000 records. This example is coded as follows:

```
          JOB          MERGE1,SCI,NREENT
          OF           INPUT01,RF1,2,SEIZE-DEV
          OF           INPUT02,RF2,2,SEIZE-DEV
          OF           INPUT03,RF3,2,SEIZE-DEV
          OF           OUTPUT,RF4,2,SEIZE-DEV
          MEM-1        13,34
          GEN          1,FIX,1,10
MSTART    MERGE        3,10C00,OUTPUT
```

Notice that no END statement need be coded since it is generated by the MERGE macro.

ISORT

The ISORT macro is used to generate an internal sort routine. The ISORT macro assumes that items to be sorted are currently in memory, and that sufficient memory is available.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
(sym)	ISORT	items,fields,length

items

The "items" field must contain the number of items to be sorted. The total number of items to be sorted must not exceed 1,000. If it is necessary to simulate the internal sorting of more than 1,000 items multiple calls on ISORT are necessary.

fields

The "fields" field must contain the total number of control fields on which the items are to be sorted.

links

The "links" field must contain the length of the items to be sorted. If the items to be sorted are of variable length, then the average length should be supplied.

EXAMPLE

Suppose the user has an array containing 500 items to be sorted on two keys. The average length of the items is 40 characters. This example is coded as follows:

```
INTSORT   ISORT              500,2,40
```

The ISORT macro does not generate an END statement and therefore may be used anywhere in a worker routine.

MATH MACROS

There are seven mathematical routines which can be simulated by means of macros as shown below.

STATEMENT FORMAT		
LABEL	OP-CD	OPERANDS
(sym)	LOG	number,digits
(sym)	SQRT	number,digits
(sym)	ABS	number
(sym)	EXP	number,digits
(sym)	ATAN	number,digits
(sym)	SIN	number,digits
(sym)	COS	number,digits

number

The "number" field must contain the total number of times the specified mathematical routine is to be evaluated.

digits

The "digits" field must contain the number of decimal places to which the expression is to be evaluated. The maximum number of places which can currently be evaluated is seven.

EXAMPLE

Suppose the user wants to represent 100 calls on a square root function for which the result must be accurate to plus of minus .0001. The example would be coded as follows:

```
SQRT            100,4
```

The number of terms to be evaluated for the purpose of the math macros are drawn from HASTINGS, CECIL JR., Approximations for Digital Computers, Princeton, 1955, Princeton University Press.

WORKING PAPER

SECTION III

USACSSEC S-3

NEW STATISTICS OUTPUT

Presented by
Leo J. Cohen Associates, Inc.

October 15, 1968

Revised October 24, 1968

334 West State Street
Trenton, New Jersey
08618
(609) 695-1488

110 North Royal Street
Suite 305
Alexandria, Virginia
22314
(703) 548-0128

WORKING PAPER

Table of Contents

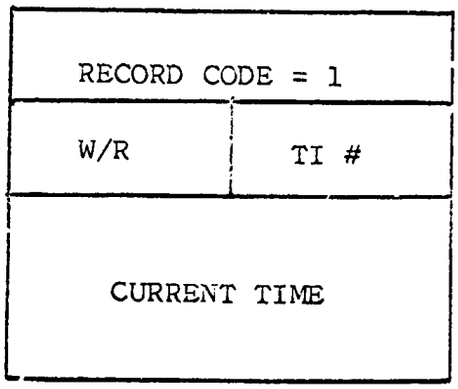
Statistics Output

Statistics Output

The following section describes the statistical records which will be written by the S-3 Simulator. These records will be written to a tape which may then be processed in order to accumulate and print any desired statistics.

W/R Creation - Figure 1

The record shown in Figure 1 will be written each time a transaction is accepted into the system by the RECEIVE statement. The record code is a 36 bit field containing the unique code. The record code of 1 indicates that this is a W/R creation record. The W/R number field is a 12 bit field and contains the number of the worker routine created. The TI number field is a 24 bit field and contains the number of the transaction item representing this worker routine's execution. The current time field contains a 72 bit double precision floating point number where the time is expressed in microseconds.



W/R = 12 bits

TI # = 24 bits

RECORD CODE = 36 bits

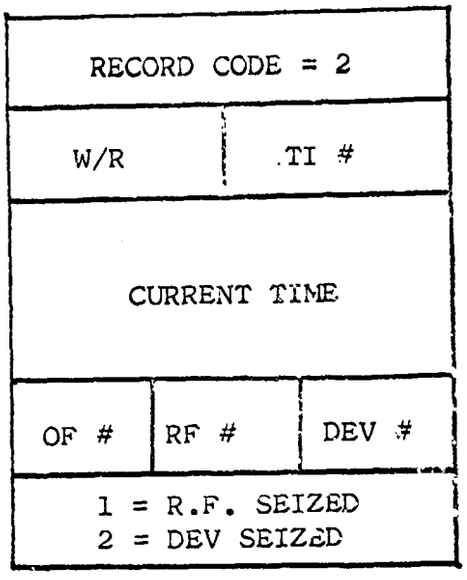
CURRENT TIME = 72 bits floating point

W/R CREATION RECORD

FIGURE 1

I/O Seize - Figure 2

The record shown in Figure 2 will be written each time a file or device is seized by means of the PERIPHERAL statement. The record code field contains the number 2 which indicates that this is an I/O Seize record. The W/R number field contains the number of the worker routine which caused this record to be generated. The TI number field contains the number of the Transaction Item associated with this record. The current time field is a double precision floating point number where the time is expressed in micro-seconds. The next word contains the ordinal file number, real file number, and device number. The last word of this record contains a code which indicates what has been seized. If the last word contains a 1 the real file specified has been seized by this statement. If the last word contains a 2 the device specified has been seized.



RECORD CODE = 36 bits

WR # = 12 bits

TI # = 24 bits

CURRENT TIME = 72 bits floating point

OF # = 12 bits

RF # = 12 bits

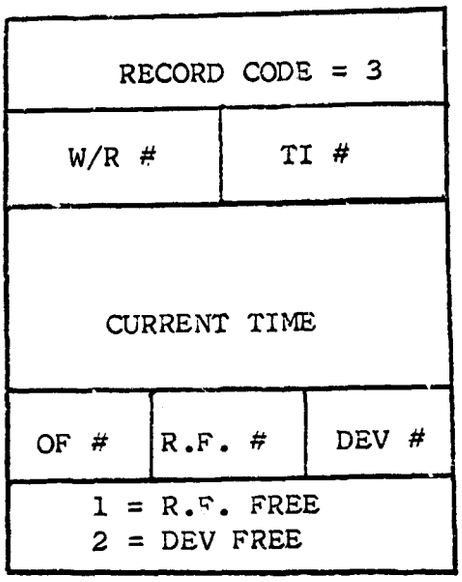
DEV # = 12 bits

I/O SEIZE RECORD

FIGURE 2

I/O Release - Figure 3

The record shown in Figure 3 will be written each time a file or device is released by means of the CLOSE statement. The record code field contains the number 3 which indicates that this is an I/O Release record. The W/R number field contains the number of the worker routine associated with this record. The TI number field contains the number of the Transaction Item associated with this record. The current time field contains a double precision floating point number where the time is expressed in microseconds. The fifth word of the record contains the ordinal file number, real file number, and device number. The last word of this record contains the code which indicates whether the file or the device has been freed by this operation. If the last word contains a 1 the real file described has been released. If the last word contains a 2 the device specified has been released.



RECORD CODE = 36 bits

W/R # = 12 bits

TI # = 24 bits

CURRENT TIME = 72 bits floating point

OF # = 12 bits

R.F. # = 12 bits

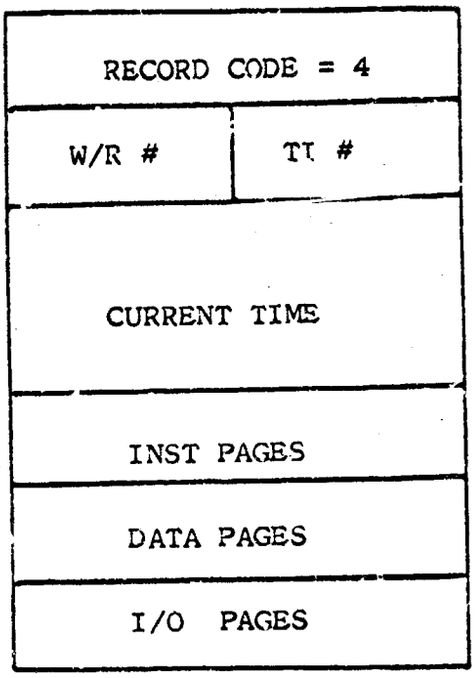
DEV # = 12 bits

I/O RELEASE RECORD

FIGURE 3

Memory Allocation - Figure 4

The record shown in Figure 4 will be written each time an ALLOCATE statement is successfully executed. The record code field contains the number 4 which indicates that this is a Memory Allocation record. The W/R number field contains the number of the worker routine associated with this record. The TI number field contains the number of the Transaction Item associated with this record. The current time field contains a double precision floating point number where the time is expressed in microseconds. The fifth word of this record contains the number of pages of memory allocated for instruction areas. The sixth word of this record contains the number of pages allocated for data areas. The seventh word of this record contains the number of pages allocated for I/O storage. Each of these last three fields contains a single precision floating point number.



RECORD CODE = 36 bits

W/R # = 12 bits

TI # = 24 bits

CURRENT TIME = 72 bits floating point

INST PAGES = 36 bits floating point

DATA PAGES = 36 bits floating point

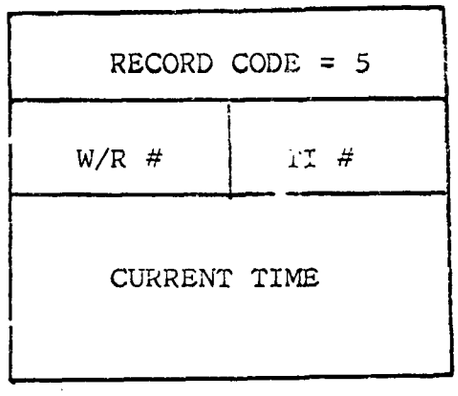
I/O PAGES = 36 bits floating point

MEMORY ALLOCATION RECORD

FIGURE 4

Memory Deallocation - Figure 5

The record shown in Figure 5 will be written each time a DEALLOCATE statement is successfully executed. The record code field contains the number 5 which indicates that this is a Memory Deallocation record. The W/R number field contains the number of the worker routine associated with this record. The TI number field contains the number of the Transaction Item associated with this record. The current time field contains a double precision floating point number where the time is expressed in microseconds.



RECORD CODE = 36 bits

W/R # = 12 bits

TI # = 24 bits

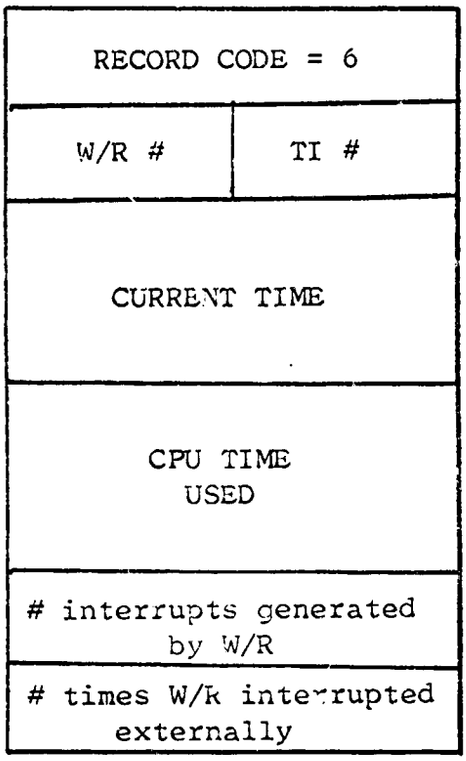
CURRENT TIME = 72 bits floating point

MEMORY DEALLOCATION RECORD

FIGURE 5

Worker Routine Termination - Figure 6

The record shown in Figure 6 will be written each time a worker routine transaction is eliminated from the system by means of the DESTROY statement. The record code field contains the number 6 which indicates that this is a Worker Routine Termination record. The W/R number field contains the number of the worker routine associated with this record. The TI number field contains the number of the Transaction Item associated with this record. The current time field contains a double precision floating point number where the time is expressed in microseconds. The CPU time used field contains a double precision floating point number where the time is expressed in microseconds. The seventh word of this record contains the number of interrupts generated by this transaction. The eighth word of this record contains the number of times this transaction was interrupted externally. The last two fields contain a full word integer value.



RECORD CODE = 36 bits

W/R # = 12 bits

TI # = 24 bits

CURRENT TIME = 72 bits floating point

CPU TIME USED = 72 bits floating point

INTERRUPTS GENERATED = 36 bits

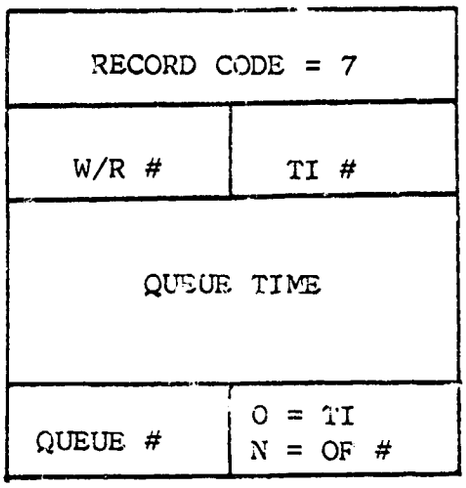
TIMES W/R INTERRUPTED = 36 bits

W/R TERMINATION RECORD

FIGURE 6

TI or IOTI SELECT - Figure 7

The record shown in Figure 7 will be written each time a transaction item or I/O transaction item is selected from a queue. The record code field contains the number 7 which indicates that this is a TI or IOTI Select record. The W/R number field contains the number of the worker routine associated with this record. The TI number field contains the number of the Transaction Item associated with this record. The queue field is a double precision floating point number where the elapsed time on the queue is described in microseconds. The eighteen right most bits in the fifth word of this record contain a code describing the item selected from the specified queue. If these bits contain a zero the item selected was a transaction item as described in the first word of this record. If these two digits contain a number, the number is the ordinal file number, for which the I/O transaction item selected from the specified queue, was generated.



RECORD CODE = 36 bits

W/R # = 12 bits

TI # = 24 bits

QUEUE TIME = 72 bits floating point

QUEUE # = 18 bits

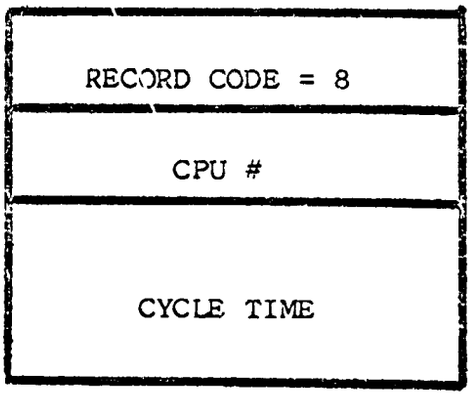
CODE = 18 bits

TI OR IOTI SELECT RECORD

FIGURE 7

O/S Cycle End - Figure 8

The record shown is Figure 8 will be written each time a CPU comes out of the cycle state. The record code field contains the number 8 which indicates that this is an O/S Cycle End record. The second word contains the number of the CPU which has just stopped cycling. The cycle time field contains a double precision floating point number where the amount of time spent cycling is expressed in microseconds.



RECORD CODE = 36 bits

CPU # = 36 bits

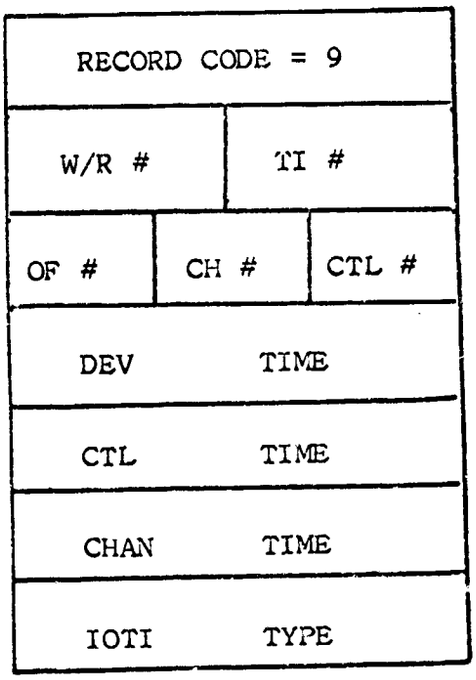
CYCLE TIME = 72 bits floating point

O/S CYCLE END RECORD

FIGURE 8

I/O Operation - Figure 9

The record shown in Figure 9 will be written each time an I/O operation is initiated by means of the I/O ADVANCE statement. The record code field contains the number 9 which indicates that this is an I/O operation record. The W/R number field contains the number of the worker routine associated with this record. The TI number field contains the number of the Transaction Item associated with this record. The third word of this record contains the ordinal file number, channel number, and control number used by this I/O operation. The fourth word of this record contains the amount of device time required for this I/O operation. The fifth word of this record contains the amount of control unit time required for this I/O operation. The sixth word of this record contains the amount of channel time required for this I/O operation. Each of these three fields are full word floating point numbers, where the time is expressed in microseconds. The last word contains the I/O Transaction Item type.



RECORD CODE = 36 bits

W/R # = 12 bits

TI # = 24 bits

OF # = 12 bits

CH # = 12 bits

CTL # = 12 bits

DEV TIME = 36 bits floating point

CTL TIME = 36 bits floating point

CHAN TIME = 36 bits floating point

IOTI TYPE = 36 bits

I/O OPERATION RECORD

FIGURE 9

Assembler Name File

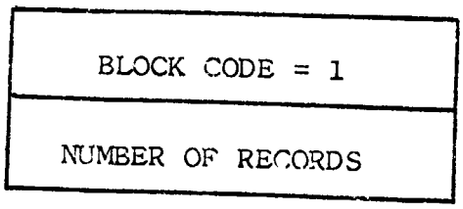
Assembler Name File

The following section describes a file which will be written by the assembler and which will be available to the statistical analysis program. In general, items of interest are supplied with names when input is prepared for the assembler. These names are then translated into numbers by the assembler for use by the simulator. As shown in the section titled "Statistics Output" the simulator refers to items of interest by number. Therefore, the assembler provides a dictionary for converting the simulator numbers back into the names supplied by the user when the input was prepared for the assembler.

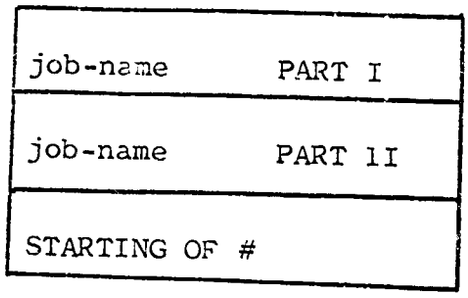
Each block in this file will be preceeded by a two word record. The first word will contain a block code describing the block which follows. The second word will contain a count of the number of records in the block. The following section describes these dictionary records.

Job Name File - Figure 1

Each record in the Job Name File will contain the name of the job left justified and right filled with blanks in the first two words. The third word will contain the starting number of the records in the ordinal file name directory for this job.



JOB NAME BLOCK HEADER



JOB NAME RECORD

JOB-NAME = 72 bits

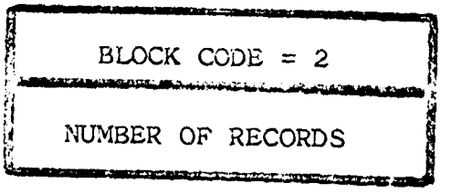
STARTING OF # = 36 bits

JOB NAME FILE

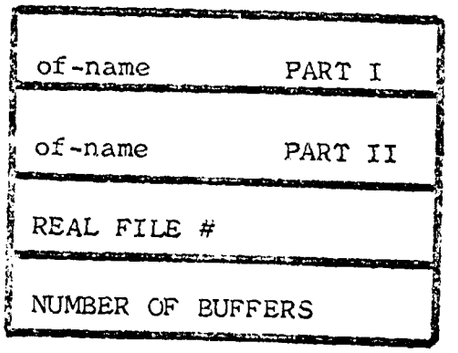
FIGURE 1

Ordinal File Name File - Figure 2

Each record in the Ordinal File Name File will contain the ordinal file name left justified and right filled with blanks in the first two words. The third word will contain the number of the real file to which this ordinal file refers. The fourth word will contain the number of buffers assigned to this file.



OF NAME BLOCK HEADER



ORDINAL FILE NAME RECORD

OF-NAME = 72 bits

REAL FILE # = 36 bits

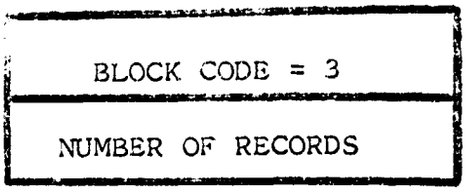
NUMBER OF BUFFERS = 36 bits

ORDINAL FILE NAME FILE

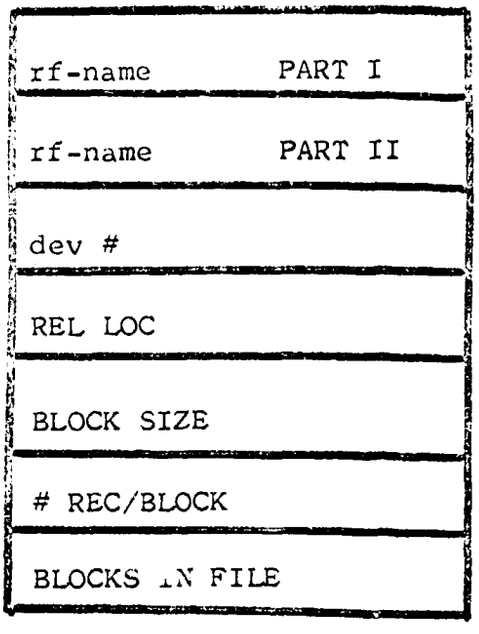
FIGURE 2

Real File Name File - Figure 3

Each record in the Real File Name File will contain the real file name left justified and right filled with blanks in the first two words. The third word will contain the number of the device on which this real file resides. The fourth word indicates the relative location of this file on the device specified in the third word of this record. The fifth word will contain the size of each block in this file expressed in characters. The sixth word will contain the number of records per block in this real file. And the last word in this record will contain the total number of blocks contained in this file.



RF NAME BLOCK HEADER



REAL FILE NAME RECORD

RF-NAME = 72 bits

DEV # = 36 bits

RELATIVE LOCATION = 36 bits

BLOCK SIZE = 36 bits

REC/BLOCK = 36 bits

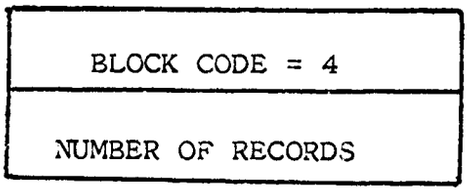
BLOCKS IN FILE = 36 bits

REAL FILE NAME FILE

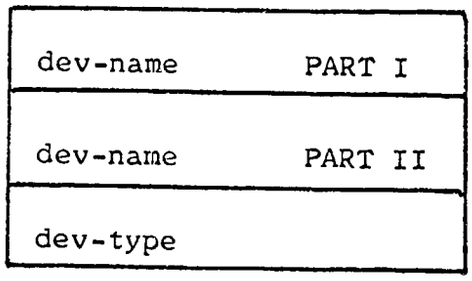
FIGURE 3

Device Name File - Figure 4

Each record in the Device Name File will contain the device name left justified and the right filled with blanks in the first two words. The third word of this record will contain a code describing the device type.



DEV NAME BLOCK HEADER



DEVICE NAME RECORD

DEV-NAME = 72 bits

DEV-TYPE = 36 bits

DEVICE NAME FILE

FIGURE 4

Channel Name File - Figure 5

Each record in the Channel Name File will contain the name of the channel left justified and right filled with blanks in the first two words. The third word of each record will contain the code describing the channel type.

BLOCK CODE = 5
NUMBER OF RECORDS

CHANNEL NAME BLOCK HEADER

chan-name	PART I
chan-name	PART II
CHANNEL TYPE	

CHANNEL NAME RECORD

CHAN-NAME = 72 bits

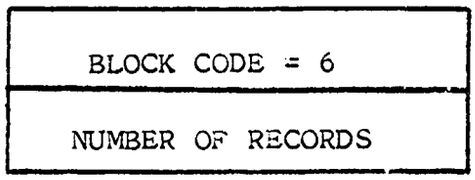
CHANNEL TYPE = 36 bits

CHANNEL NAME FILE

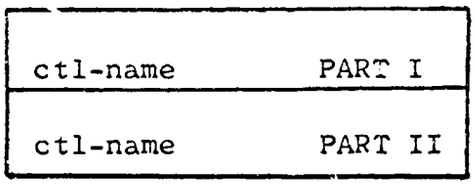
FIGURE 5

Control Name File - Figure 6

Each record in the Control Name File will contain the control name left justified and right filled with blanks in the first two words.



CONTROL NAME BLOCK HEADER



CONTROL NAME RECORD

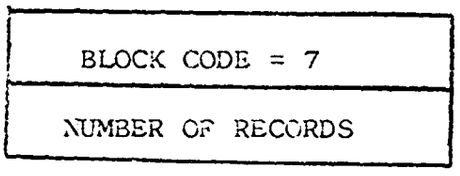
CTL-NAME = 72 bits

CONTROL NAME FILE

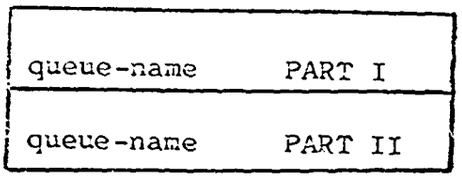
FIGURE 6

Queue Name File - Figure 7

Each record in the Queue Name File will contain the name of the queue left justified and right filled with blanks in the first two words.



QUEUE NAME BLOCK HEADER



QUEUE NAME RECORD

QUEUE-NAME = 72 bits

QUEUE NAME FILE

FIGURE 7

CPU Name File - Figure 8

Each record in the CPU Name File will contain the name of the CPU left justified and right filled with blanks in the first two words. The third word will contain the size of the logical data unit in bits, The fourth word will contain the number of characters per logical data unit.

BLOCK CODE = 8
NUMBER OF RECORDS

CPU NAME BLOCK HEADER

CPU-NAME PART I
CPU-NAME PART II
LDU SIZE IN BITS
CHAR/LDU

CPU-NAME RECORD

CPU-NAME = 72 bits

LDU SIZE = 36 bits

CHAR/LDU = 36 bits

CPU NAME FILE

FIGURE 8

Memory Name File - Figure 9

Each record in the Memory Name File will contain the name of the Memory left justified and right filled with blanks in the first two words. The third word will contain the size of the memory access unit. The fourth word will contain the memory size in memory units. The fifth word will contain the page size in memory units.

BLOCK CODE = 9
NUMBER OF RECORDS

MEMORY NAME BLOCK HEADER

MEM-NAME PART I
MEM-NAME PART II
MEMORY ACCESS UNIT
MEMORY SIZE
PAGE SIZE

MEMORY NAME RECORD

MEM-NAME = 72 bits

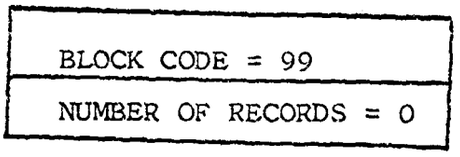
MEM ACC UNIT = 36 bits

MEM SIZE = 36 bits

PAGE SIZE = 36 bits

MEMORY NAME FILE

FIGURE 9



END OF NAME FILE

FIGURE 10